80/310/FDIS

FINAL DRAFT INTERNATIONAL STANDARD
PROJET FINAL DE NORME INTERNATIONALE

	Project number / Numéro de projet **IEC 61162-401 Ed.1**	
	IEC/TC or SC CEI/CE ou SC **TC 80**	Secretariat / Secrétariat **United Kingdom**
☒ Submitted for parallel voting in CENELEC / Soumis au vote parallèle au CENELEC	Distributed on / Diffusé le **2001-08-24**	Voting terminates on / Vote clos le **2001-10-26**
Also of interest to the following committees / Intéresse également les comités suivants	Supersedes document / Remplace le document 80/261/CDV - 80/295/RVC	

Functions concerned / Fonctions concernées

☐ Safety / Sécurité	☐ EMC / CEM	☐ Environment / Environnement	☐ Quality assurance / Assurance de la qualité

INTERNATIONAL ELECTROTECHNICAL COMMISSION — COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

Title
Maritime navigation and radiocommunication equipment and system - Digital interfaces - Part 401: Multiple talkers and multiple listeners - Ship systems interconnection - Application profile

Titre

ATTENTION
VOTE PARALLÈLE
CEI – CENELEC

L'attention des Comités nationaux de la CEI, membres du CENELEC, est attirée sur le fait que ce projet final de Norme internationale est soumis au vote parallèle. Un bulletin de vote séparé pour le vote CENELEC leur sera envoyé par le Secrétariat Central du CENELEC.

ATTENTION
IEC – CENELEC
PARALLEL VOTING

The attention of IEC National Committees, members of CENELEC, is drawn to the fact that this final Draft International Standard (DIS) is submitted for parallel voting. A separate form for CENELEC voting will be sent to them by the CENELEC Central Secretariat.

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

____

MARITIME NAVIGATION AND RADIOCOMMUNICATION EQUIPMENT AND SYSTEMS – DIGITAL INTERFACES –

Part 401: Multiple talkers and multiple listeners – Ship systems interconnection – Application profile

FOREWORD

1) The IEC (International Electrotechnical Commission) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of the IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, the IEC publishes International Standards. Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. The IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.

2) The formal decisions or agreements of the IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested National Committees.

3) The documents produced have the form of recommendations for international use and are published in the form of standards, technical specifications, technical reports or guides and they are accepted by the National Committees in that sense.

4) In order to promote international unification, IEC National Committees undertake to apply IEC International Standards transparently to the maximum extent possible in their national and regional standards. Any divergence between the IEC Standard and the corresponding national or regional standard shall be clearly indicated in the latter.

5) The IEC provides no marking procedure to indicate its approval and cannot be rendered responsible for any equipment declared to be in conformity with one of its standards.

6) Attention is drawn to the possibility that some of the elements of this International Standard may be the subject of patent rights. The IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 61162-401 has been prepared by IEC technical committee 80: Maritime navigation and radiocommunication equipment and systems.

The text of this standard is based on the following documents:

FDIS	Report on voting
80/XX/FDIS	80/XX/RVD

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 3.

The special typographical conventions and nomenclature used in this standard are defined in IEC 61162-400, annex A.

Annexes A, B and C form an integral part of this standard. Annex D is for information only.

The committee has decided that the contents of this publication will remain unchanged until June 2005. At this date, the publication will be

- reconfirmed;
- withdrawn;
- replaced by a revised edition, or
- amended.

INTRODUCTION

International Standard IEC 61162 is a four-part standard which specifies four digital interfaces for applications in marine navigation, radiocommunication and system integration.

The four parts are:

IEC 61162-1	Single talker and multiple listeners
IEC 61162-2	Single talker and multiple listeners, high speed transmission
IEC 61162-3	Multiple talkers and multiple listeners – Serial data instrument network
IEC 61162-4	Multiple talkers and multiple listeners – Ship systems interconnection.

Part 4 of the standard is sub-divided into a number of individual standards with part numbers in the IEC 61162-400 series. A full reference to part 4 can be found in IEC 61162-400, clause 4.

This part of the standard, IEC 61162-401: A-profile specification, defines the application functionality and its implementation in an application layer protocol.

Relationship with the other parts of the IEC 61162 series of standards is defined in annex B to IEC 61162-400.

MARITIME NAVIGATION AND RADIOCOMMUNICATION EQUIPMENT AND SYSTEMS – DIGITAL INTERFACES –

Part 401: Multiple talkers and multiple listeners – Ship systems interconnection – Application profile

1 Scope

1.1 General

IEC 61162-4 series specifies a communication system for use in integrated ship control systems.

IEC 61162-400 defines the overall functional scope for the communication system.

1.2 Application profile

This part of IEC 61162 describes the application profile (A-profile – corresponding to ISO-OSI layers 5 to 7 [ISO 7498]) of the communication protocol which is the basis for the communication system. It relies on the realization of layers 1 to 4 (the T-profile) as described in part 410.

The description of the A-profile is in terms of services offered to the application using the protocol and of message contents and sequences used to realize these services.

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of IEC 61162. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of IEC 61162 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of IEC and ISO maintain registers of currently valid International Standards.

IEC 61162-400, *Maritime navigation and radiocommunication equipment and systems – Digital interfaces – Part 400: Multiple talkers and multiple listeners – Ship systems interconnection – Introduction and general principles*

IEC 61162-410, *Maritime navigation and radiocommunication equipment and systems – Digital interfaces – Part 410: Multiple talkers and multiple listeners – Ship systems interconnection – Transport profile requirements and basic transport profile*

IEC 61162-420, *Maritime navigation and radiocommunication equipment and systems – Digital interfaces – Part 420: Multiple talkers and multiple listeners – Ship systems interconnection – Companion standard requirements and basic companion standards*

IEEE 754: *IEEE Standard for Binary Floating-Point Arithmetic*

ISO/IEC 8859-1, *Information technology – 8-bit single-byte coded graphic character sets – Part 1: Latin alphabet No. 1*

ISO/IEC 10646-1, *Information Technology – Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane*

RFC 2500:1999, *Internet Official Protocol Standards – Internet Activities Board standard*

NOTE RFC (request for comments) is a document issued by the Internet engineering task force (IETF), the International standardization body for the Internet, that describes a part of the Internet protocol. Some RFCs are accepted as official Internet standards and listed in the "Internet Official Protocol Standards" itself an RFC.

3 Definitions

This clause is divided into definition of terms (terms), definition of abbreviations (abbreviations), definitions of nomenclature (conventions), definition of data types (data types) and definition of literal formats. Other definitions valid for this part of IEC 61162 are contained in part 400 of this standard.

3.1 Terms

For the purpose of this part of IEC 61162, the following terms apply:

3.1.1
anonymous broadcast (ABC)
a broadcast service where the sender does not know to which MAU it is sending data. Similarly the listener may not know which sender it should listen for

3.1.2
array
a linear indexed sequence of identical data types. The index runs from zero and upwards. Arrays can have variable lengths (with a fixed upper limit) or fixed lengths. The difference between these two types is normally only visible during transmission between modules where the real length of a variable length array is transmitted as an attribute

3.1.3
bit order
this standard numbers bits in an octet from zero to seven. Bit seven is the most significant bit; bit zero the least significant

3.1.4
character
this standard provides two mechanisms for the transmission of characters:

a) an 8-bit character based on ISO/IEC 8859-1 (also called ISO Latin-1). This set covers most national alphabets based on the Latin letters;
b) a 16-bit character based on ISO/IEC 10646-1. This standard specifies the use of the 16-bit form Universal Character Set 2 (UCS-2) which covers most of the commonly used character sets in the world

NOTE 1 Later revisions of the standard may also support 32-bit characters.

NOTE 2 Any reference to *character* in this standard implies the 8-bit character if not otherwise stated.

3.1.5
companion standard
these are the mechanisms to define and describe how the A-profile services are used to implement some application functions and interfaces (see IEC 61162-420)

3.1.6
connection
an association between two interfaces or two MCPs, one each on a server and a client MAU. A connection must be established before transactions can be activated

There is also a connection between each pair of LNAs and between each MAU and its LNA. Although similar in concept, they are not directly associated with the activation of transactions.

3.1.7
data type
this standard defines a set of data types that have a machine and T-profile independent interpretation. The types cover, for example characters, signed and unsigned integers and floating point containers as well as some other derived types. The data types are defined in 3.4

3.1.8
format string
a text string that defines the structure of certain data records that are transmitted via the T-profile. It also contains information about the function supported by data objects. See clause 8.4.3.5 for more information

3.1.9
interface
a collection of MCPs (references to data objects) implemented on a server or client MAU which behaves similarly to MCPs in that operations on interfaces effect all MCPs in the interface

3.1.10
IP – internet protocol
references in this standard refer to the protocol defined in RFC 2500. Part 410 of IEC 61162 specifies a T-profile using version 4 of the internet protocol (IPV4), but additional parts may in the future define T-profiles using newer versions of the protocol

3.1.11
IPV4 – internet protocol version 4
is currently the most used version of the internet protocol (IP). See IEC 61162-410 for more details

3.1.12
magic number
The first field of all messages which will be common to a group of messages. It can be used by the protocol software to identify and verify messages and message boundaries. See annex A

3.1.13
maximum message size
this protocol is based on the exchange of messages between various modules. The T-profile may impose limits on the maximum size of these messages. See clause 4 for more information

3.1.14
message
one of the basic components of this standard. They are collections of information items with a defined format that are exchanged between various modules to achieve some service or part service. The message formats are defined in clause 8. The exchange patterns are defined as sequence diagrams in clause 7

3.1.15
module
entities (programs or host computers) between messages that are sent which are either an LNA (application independent network communication manager) or a MAU (the application program module)

3.1.16
network bit order
the A-profile definition is resolved to the octet level. The ordering of bits within each octet is defined by the T-profile in use. See clause 4 for more information

3.1.17
network octet order
all multi-octet entities (messages or fragments of messages) are defined to be transferred in a T-profile and host computer independent format. This format is defined in 8.4.3.5

3.1.18
null MCP
an MCP associated with the MAU itself. It will have a reference code of zero. It can be manipulated similarly to a normal MCP through the MAU control data object (see 6.5) and it can also be used to transfer information to MAU call back routines (see 5.2.4). The null MCP cannot be connected to from remote MAUs and it is not associated with any interface

3.1.19
octet
the smallest information entity that the A-profile protocol considers which consists of 8 bits transmitted by the T-profile in some specific sequence. This standard requires that a group of octets can be transmitted as a message and that the received message has the same sequence of octets as the sent message and that each received octet has the same bit pattern as the sent octet. The protocol does not specify any particular ordering of bits within octets as long as it is consistent over the network

NOTE One entity of the Boolean type can be represented as a bit within an octet, but any number of Boolean entities will be transmitted as an integral number of octets.

3.1.20
record
a sequence of different or identical data types given a fixed definition and an associated type name

3.1.21
sequence diagram
a representation of how messages are transferred between the modules of the protocol system. The format is defined in 3.3.3

3.1.22
session
the term is used to identify a connection from a client MAU to a server MAU. It represents one unbroken connection where the LNAs guarantee that no other client MAU has replaced the originally connecting MAU. The death of a client MAU will be reported to the server MAU as a closed session

3.1.23
T-profile network
a collection of network nodes (host computers) that can communicate with each other following this standard. The concept is defined in IEC 61162-410.

This standard does not specify how nodes on two different T-profile networks shall communicate

NOTE The standard does, however, specify that one node shall be able to be connected to more than one T-profile network at one time and this can be used to develop gateway nodes.

3.1.24
TCP/IP – transmission control protocol / internet protocol
for version 4 of the internet protocol (IPV4) is defined in RFC793. See IEC 61162-410 for more details

3.1.25
transaction
the term is used to identify an exchange of data between two MAUs that is related to one initial request from the client. It can consist of from zero (non-acknowledged write) to an unlimited number (subscriptions) of messages from the server

3.1.26
version number
in this standard, it is 4.0 (major number is 4, minor is 0). Earlier version codes have been used by MiTS (see IEC 61162 and IEC 61162-400)

3.2 Abbreviations

Table 1 lists important abbreviations used in this standard. The third column gives a reference to the clause where a definition of the term can be found.

Table 1 – Abbreviations

Term	Description	Reference
ABC	Anonymous broadcast	3.1.1
Ack	Acknowledge	Figures & text
IPC	Inter-process communication	11.1
IP	Internet protocol	3.1.10
IPV4	Internet protocol version 4	3.1.11
LNA	Local network administrator	Part 400
MAU	MiTS application unit (application program module)	Part 400
MCP	MAU connection point	5.3
MTU	Maximum transmission unit	4.1
Obj	Object	Figures & text
Req	Request	Figures & text
RLNA	Remote LNA (in sequence diagram)	3.3.3
RMAU	Remote MAU (in sequence diagram)	3.3.3
TCP/IP	Transmission control protocol/internet protocol	3.1.24
UCS	Universal character set	3.1.4

3.3 Conventions

3.3.1
use of type face
each description element has its own conventions for use of type face. In this standard, the following conventions apply:

a) use of `courier` type face indicates either some defined protocol data type (see next item) or another protocol identifier (e.g. a message code, see 3.3.2);

b) use of the extension `_m` in a word type set in `courier` means that it references one of the defined protocol data types. The types are defined in 3.4;

c) *italics* are normally used when a word or term references a label or identifier in an associated table or figure.

3.3.2
message definition template
this standard describes a message based protocol. Each message will be described as an ordered list of fields. An example is shown in table 2.

Table 2 – Example of message definition

Octet #	Data type	Field description
0 – 1	`word16_m`	`MV4_MAGIC`
2 – 3	`word16_m`	*type*, message type
4 – 5	`word16_m`	Priority
6 – 7	`word16_m`	*length*, message length (= x+1 or y+1)
8 – x		message body necessary
– y		message body sometimes required

The message definition has three columns.

a) The first column gives the octet numbers used by a field, numbered from the first to the last octet inclusive of the field. The numbering starts with `0` at the first octet of the message. The letters x, y, etc. are used to indicate the last byte number in a variable length field. Arithmetic expressions may also be indicated in this column. Arithmetic expressions are enclosed in parenthesis.

b) The second column gives the data type of the field (see 3.1.7 and 3.4.2).

c) The third column describes the contents of the field. Identifiers in `courier` show constant values. The value of these can be found in the annexes. Expressions typeset in *italics* are described in the text following the message definition.

The physical message consists of the entities defined in the table transmitted as a sequence of (multi-octet) quantities. A message may consist of one part that is always transmitted and one part that is sometimes transmitted. The limit between these two parts is shown by a thick row divider line in the table

3.3.3
sequence diagrams
message sequences are described in diagrams. An example is shown in figure 1

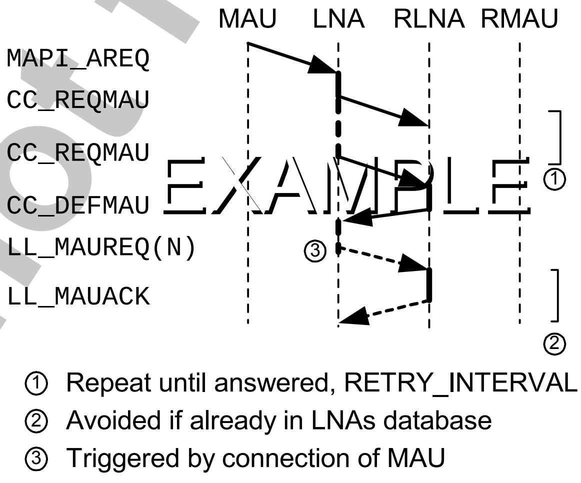
MAU
LNA
RLNA
RMAU
MAPI_AREQ
CC_REQMAU
CC_REQMAU
CC_DEFMAU
LL_MAUREQ(N)
LL_MAUACK
EXAMPLE
①
②
③
① Repeat until answered, RETRY_INTERVAL
② Avoided if already in LNAs database
③ Triggered by connection of MAU

Figure 1 – Sequence diagram example

The following graphical elements are used:

– each **communicating module** is represented by a vertical dashed line. The upper list of names identify the modules. The same module list will be used in all diagrams (the modularization is defined in IEC 61162-400):

1) **MAU** is the local application unit communicating with its LNA;

2) **LNA** is the local communication manager communicating with MAU and RLNA;

3) **RLNA** is the remote communication manager communicating with LNA and RMAU;

4) **RMAU** is the remote application unit communicating with its RLNA;

NOTE When two MAUs which are connected to the same LNA are communicating, the LNA – RLNA communication sequence may depend on the implementation. However, it is recommended that the implementation uses the standard message sequence and defines some kind of LNA internal loop-back interface for internal messages.

– **time** runs from top to bottom. If needed, timing information may be shown explicitly on the right side as comments, otherwise no time scale is given;

– **messages** are shown as arrows between the vertical lines. The arrows are pointing from the sender to the receiver, and downwards. Dashed arrows (as in comment (2)) indicates messages that are sent only under certain conditions;

– **message code** is shown on the far left side, aligned with the arrow start point. The code is associated with a message structure that is defined in clause 8. Two special short-hand notations are used:

  1) a trailing (N) in parenthesis defines a message with a negative acknowledgement,

  2) a trailing (C) in parenthesis specifies two alternate messages. Explanation is given in the text;

– **a thick solid vertical line** along the module line is used if the reception of a message causes an “immediate” sending of one or more new messages. This thick line should be interpreted as showing that the connected operations are executed in the same control thread and within a relative short time frame;

– **a thick dashed vertical line** is used if the new message is sent after a (possibly) significant time, for example after a timer expires (as in comment (1)) or as a result of another internal event (as in comment (3)),

– **sequence indicators** to group one or more messages or other graphical elements (usually as a line on the right hand side) associated with a comment reference. This means that the comment applies to the group of graphical elements (as example for comment (1) and (2));

– **comment references** as numbers enclosed in circles. The number will refer to a comment either in the drawing itself or in the following text, with the number in parenthesis.

The graphical elements are supplemented by other textual explanation where necessary.

3.4 Protocol data types

3.4.1
general
this clause defines the data types used by this standard. Data type is defined as a data container which is capable of containing information with a certain precision. Any implementation of this protocol shall make sure that the host computer’s or the T-profile’s representation of these data types have at least the precision defined in the following clauses

3.4.2
basic type
for numerical and textual values, the standard makes references to the set of basic types described in Table 3. The standard will use the names listed under the *type* column to identify any of the basic types. The format column specifies the format string encoding for the relevant type and the short format column lists the one-character code that can be used to identify the type, for example in software libraries (see 8.4.3.5).

Table 3 – Basic entities

Type	Format	Short format	Description
`bool_m`	`b`	`l`	1 bit Boolean (0 = FALSE, 1 = TRUE)
`char8_m`	`c8`	`c`	8 bit character [ISO/IEC 8859-1]
`char16_m`	`c16`	`C`	16 bit character [ISO/IEC 10646-1], UCS-2
`word8_m`	`w8`	`b`	8 bit unsigned integer
`word16_m`	`w16`	`w`	16 bit unsigned integer
`word32_m`	`w32`	`W`	32 bit unsigned integer
`word64_m`	`w64`	`U`	64 bit unsigned integer
`int8_m`	`i8`	`t`	8 bit 2's complement integer
`int16_m`	`i16`	`I`	16 bit 2's complement integer
`int32_m`	`i32`	`I`	32 bit 2's complement integer
`int64_m`	`i64`	`H`	64 bit 2's complement integer
`float32_m`	`f32`	`f`	32 bit floating point [IEEE 754]
`float64_m`	`f64`	`F`	64 bit floating point [IEEE 754]

NOTE 1 The `char8_m` data-type can be both signed and unsigned. This is dependent on the implementation. It shall, however, be guaranteed true to a relevant machine's representation of an 8-bit ISO/IEC 8859-1 character.

NOTE 2 Expect an extension of this list to include higher precision entities, for example 32-bit characters, 128-bit integers and 128-bit floating point. It is recommended that coding practices take this into consideration.

3.4.3
array

arrays are sequences of basic types, unions or records (not arrays). Arrays are variable length or fixed length, with a special type of variable length array called an opaque block:

a) **arrays**: this is an indexed list of element types. It is defined by element type and array length. Elements are numbered from zero and up. The length is constant and all elements are transmitted on the network;

b) **variable length arrays** (var-array): this is an array with a variable length. It is defined by element type, maximum length and the type of an unsigned integer used to contain the length of the transmitted array. Only the length field itself and the number of elements specified in the length field are transmitted on the network.

c) **opaque block:** this is a special case of the variable length array that is used to support more complex formats that cannot be represented in the current formatting strategy. An opaque block is a variable length array of unsigned octets (`word8_m`) that must be further interpreted by a specified information definition.

In this part of IEC 61162, only fixed length arrays are used. They are indicated by putting the array length in square brackets after the array type name, for example `word16_m[32]`.

A special case of the array is the null terminated string. This is a sequence of characters terminated by the null character. The maximum length of the string is always defined. This type is only used in the A-profile definition and is not accessible to the application programmer

3.4.4
record
is an ordered list of basic types, other records or arrays. All elements or arrays are transmitted. It is defined as an ordered list of elements and/or arrays

Records are not used in this part. They are only used by application programmers. Messages can be regarded as a special case of records. Messages are defined in clause 8.

3.4.5
union
is similar to a record, but with the definition of an additional unsigned integer used to index the element or array that is actually transmitted in one given transfer. Only one element or array in the basic record can be transmitted. The union is defined as a record, but with the additional index type

Unions are not used in the A-profile definition. They are available for application programmers

3.4.6
derived types
this part of the standard also specifies a set of derived types that are used to contain certain special protocol codes. These are described in Table 4. The format column specifies the format string encoding for the relevant type (see 8.4.3.5). Note that the `Ns` format represents the special array type called "null terminated string" (in this text references to as `string_m`)

Table 4 – Derived number formats

Type	Format	Description	Reference
address_m	(w16[w8:48]w8)	Network address for a T-profile network	9.2
nodenumber_m	w32	Network node number in a T-profile network	9.2
mauid_m	w16	LNA specific identifier for a local MAU	9.3
mauname_m	32s	MAU name	9.3
mcid_m	w16	Data object identifier code	9.4
mciname_m	32s	Data object name for interface MCP	9.6
ifname_m	32s	Interface name	9.7
password_m	8s	Password, fixed length array of 8 char8_m	9.11
word16_m	w16	Identifier for remote MAU-LNA session	9.8
string_m	Ns	Null terminated string of maximum length N (inc. null)	0
word32_m	w32	Identifier for a transaction	9.9

3.5 Non-protocol types

Some data types are used in this standard to refer to various data entities that need to be implemented in the API or in the LNA, but which do not occur in protocol messages. The table below gives an overview of these.

Table 5 – Non-protocol data types

Type	Description	Reference
val_m	A general integral value, either integer or reference to structure	5.4.3
eventh_m	A reference or reference to a call-back function	5.4.3
pack_m	A reference to a data marshalling function	5.4.3
priority_m	Priority (PT_LOW, PT_NORMAL, PT_URGENT)	5.4.3

The data types are defined based on semantics of the C language and may be more or less difficult to map to other languages

3.6 Literals

This standard uses literals as defined in annex A of IEC 61162-400. The principles are summarized below.

3.6.1
numbers
decimal and hexadecimal (prefixed with `0x`) numbers are used. Decimal numbers can be signed and/or floating point

3.6.2
characters and strings
single characters are enclosed in single quotes, strings in double

3.6.3
Boolean
the Boolean values `TRUE` (numeric value 1) and `FALSE` (numeric value 0) are used

3.6.4
symbolic constants
annexes A to C define symbolic constants used in this standard. All references to these constants in this standard are by the constant's symbolic name in `courier` typeface

4 Dependence on T-profile

4.1 General

This part of the standard is written in a form that makes it independent of the T-profile. The principle is that different T-profiles can be used, dependent on the application area and corresponding requirements. The standard principles for the definition and construction of a T-profile are contained in IEC 61162-410.

This clause gives an overview of the assumptions that this part has been based upon and some of the deviations that can be allowed.

4.2 LNA-LNA or MAU-LNA communication

The LNA is only required to use the T-profile for LNA-LNA communication. However, it is suggested that the LNA also uses the same logical structure (TP-network, network node number) for MAU-LNA communication (see 6.4.1.2).

4.3 Services required

The T-profile is assumed to provide node connection and data transport services as described in IEC 61162-410 of this standard. Briefly summarized, these services are:

a) define and establish one or more T-profile networks where nodes are individually identifiable by network node number;
b) establish reliable (and possibly redundant) communication links for messages between any two nodes on the network;
c) establish listening and transmitting multi-cast ports that can send or receive messages between and to all nodes on a T-profile network;

d) detect and report errors on a connection, including loss of redundancy;

e) send and receive reliable messages on two priority levels (PT_URGENT and PT_NORMAL);

f) send and receive unreliable broadcast (or multicast) messages on the PT_URGENT and PT_NORMAL priority level.

This standard enables communication between nodes on one T-profile network.

4.4 Services required, but not used by the A-profile

The T-profile shall in addition to the services listed in the previous subclause provide two further management-oriented services:

a) physical network management;

b) time distribution.

These services are required for the integrated control system, but will not be used to implement A-profile services as described in this part of the standard. These services are assumed to be handled independently of the implementation of the A-profile in the host computer.

4.5 Variable quality of service attributes

The A-profile does not necessarily need all services described above, nor does it need to have the same quality of service for all variants of T-profiles. This subclause describes allowable deviations from the assumed quality of service and the consequences for A-profile functionality.

4.5.1 Reliability and safety

The T-profile is in principle assumed to give reliable delivery of certain types of data, also in face of partial loss of communication capability. Furthermore, it is assumed that no single error in the lower level transport functionality shall disrupt the delivery of data. This normally requires some form of redundancy, retransmission capabilities and sufficient self-checking of T-profile transmission mechanisms.

The A-profile will function without this level of reliability. However, the basic safety related properties of the communication system will be severely affected by lower service quality in these areas. For non-safety-related integration tests or non-critical data acquisition systems, it will normally be sufficient to have a lower integrity T-profile service, for example by using the worldwide Internet. For safety related systems, this will not be sufficient.

It is necessary to consider the application area for the system before a decision is made to allow a lower quality T-profile to be used. This must also be clearly documented.

For broadcast messages, the A-profile and the applications are required to assume that messages can be lost or duplicated. Both losses and duplication may impact on the performance of the system. The T-profile should reduce the possibilities for any of this to happen.

4.5.2 Real time response

The T-profile is required to provide three priority levels: PT_URGENT, PT_NORMAL and PT_LOW. Higher priority traffic shall have precedence over lower priority traffic. In addition, it is assumed that the T-profile can supply some kind of response guarantees for each of the priority levels.

The three priority levels are sufficient to give some level of priority support even on simple T-profiles without quality of service support on the transport level, for example as for Internet version 4 over Ethernet. The use of individual buffers on the data link level and prioritized scheduling of messages from the buffers should give sufficient priority control for most purposes. However, actual parameters must be determined based on implementation characteristics and load figures.

4.5.3 Support for stream data

Stream data interfaces are not required by the A-profile specification. To use stream data interfaces, the MAUs have to establish a direct stream between them. The establishment of such streams is defined in the companion standards. The implementation of the stream is defined by the T-profile.

4.5.4 Maximum and minimum message lengths

This standard does not enforce any upper limit on message sizes. However, the underlying T-profile may define some limits that will be propagated to the A-profile, for example:

a) for urgent messages, the message length will normally be limited to the maximum transmission unit (MTU) of the transport protocol in use;
b) for some transport protocols there may be general message length restrictions due to protocol particulars, for example for message-oriented transfers;
c) some implementations may limit the message length to increase efficiency.

The maximum message length, if enforced by the implementation, shall be readable by the application through specified API calls (see service definitions in IEC 61162-410).

The A-profile will be designed to accommodate message sizes as low as 512 bytes of payload data. This constant is defined as `MV4_MES_MIN` in annex C. Note that this will reduce the available payload also for application programs.

4.6 Congestion and flow control

The A-profile will use flow control mechanisms in the T-profile to inhibit further transmissions on a communication link if it is congested. The flow control shall only be used on low and normal priority links. The flow control will be applied on the session level (see 5.2.4). A session can be looked at as an association between two MAUs.

In case of congestion, the A-profile shall queue non-transmitted messages until the T-profile is able to receive new outgoing messages. The implementation shall as far as possible avoid that messages are cancelled due to congestion. However, an application error that makes the MAU continue to send messages on a congested link may be handled by closing the session completely.

5 Functional requirements for MAU

5.1 General overview

The context diagram for the MAU is shown in Figure 2 The MAU uses its local LNA for all communication with the network. It may be equipped with a diagnostics console, but the functionality of this is not defined in this standard.

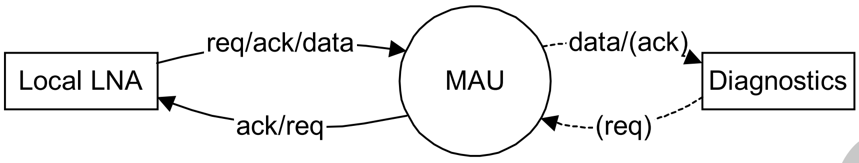

Figure 2 – MAU context diagram

The MAU is divided into the application specific part and the application programmer's interface (API). The latter part implements the application independent part of the MAU and takes care of mapping the application services to and from the relevant LNA-MAU messages. The implementation of the application services in the LNA are discussed in clause 6.

This clause defines the services that this protocol makes available to the application programs (MAU). Most of these services are implemented as a message to the LNA, possibly with some form of acknowledgement being expected later. In this case, the message name is included in the service definition. The protocol can then be determined by looking up the corresponding sequence diagram.

Other services require the use of an application program interface (API) with more functionality than simply a translation from subroutine call to protocol message. These services are not referenced to a particular message.

All services are required and must be implemented by all open APIs supporting this protocol. Clause A.7 of IEC 61162-400 gives an overview of the service definition format and how to convert it into an API.

5.2 MAU configuration management

The services defined in this subclause are used to manipulate MAU state, from connection to LNA to removal. The services are based on the state transition diagram shown in Figure 3.

The states are:

- **`MAU_DEFINED`**: a call to the service `MauInitialize` defines the MAU's context and defines it as a local non-connected application unit. This state can be used to manipulate attributes of connection points or define connection points, but no connections can be established;
- **`MAU_OPENING`**: a connection attempt to the LNA is initiated by a call to `MauOpen`. The MAU will stay in this state until the connection is established, it times out or an unrecoverable network error occurs. The application program must wait for one of these events or explicitly request a close (`MauClose`),
- **`MAU_OPEN`**: when the connection to the LNA is established, the MAU enters the open state. At this stage, the MAU is defined in the network and has been assigned a set of standard services accessible by the LNA and other MAUs in the system. Connection points can now be established, and when successfully opened, used for transactions,
- **`MAU_CLOSED`**: the connection to the LNA has been requested to close by an external unit (LNA or another MAU) or by the application itself calling `MauClose`. The application program waits for the `EV_LNA_RESTART` signal before it reopens the MAU-LNA connection;

NOTE The external request can usually be denied by the MAU, in which case the MAU stays in the open state. The kill signal cannot be denied.

- **`MAU_ERROR`**: a non-recoverable communication error occurred. New communication parameters must be established before a new attempt can be made.

•

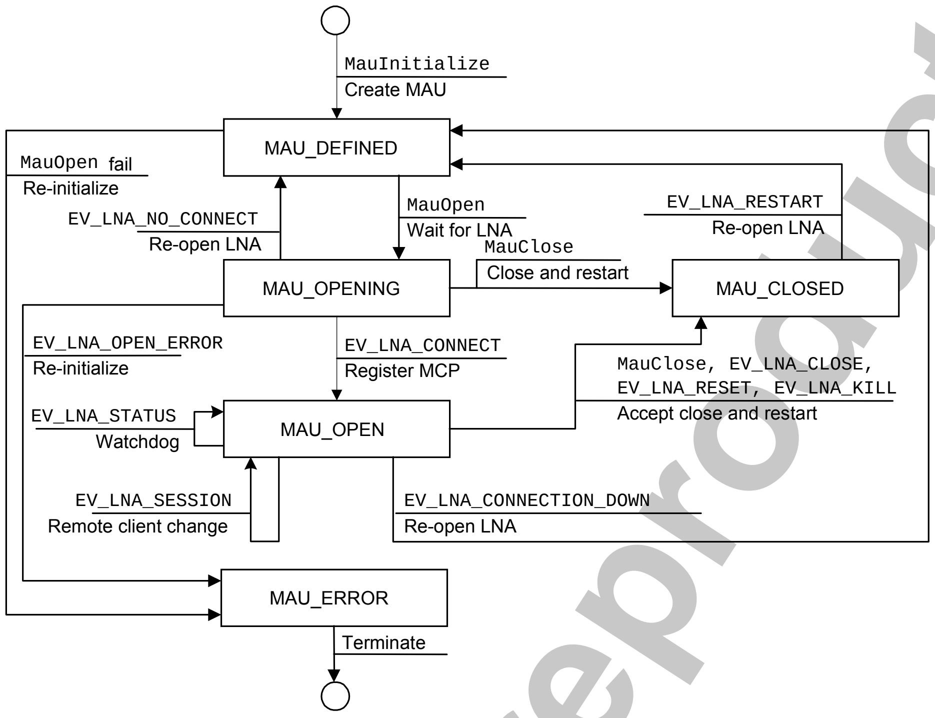

Figure 3 – MAU state diagram

Note that the MAU has an associated MCP (the null MCP, see 6.5) that can be used to control and monitor some of the connection related attributes of the MAU. The *last error* attribute for the null MCP will return any error details associated with a MAU level message exchange.

5.2.1 Define MAU parameters (`MauInitialize`)

NOTE This call is used to define certain parameters that are used when the MAU is later instantiated.

Pre-condition: OS and communication sub-system initialized. A communication descriptor for MAU-LNA communication has been created.

Post-condition: system ready to open a communication channel to the LNA or an unrecoverable error occurred.

Input parameters:

a) communication descriptor: a reference to the communication channel used between MAU and its LNA, usually provided through a T-profile API (TLI);
b) maximum number of data objects: this is used to allocate memory for descriptor tables. May be unnecessary in some implementations;
c) MAU name: logical name of application module. It is possibly to let the LNA define a unique name by using a default MAU name;
d) MAU description: a descriptive string,
e) MAU password: used to protect against unauthorized access to MAU control functions;
f) a call-back routine for state changes as shown in the figure,
g) an optional up-call routine (possibly with other semantics than a clean call-back) which is used to report internal errors and warnings from the API.

Output parameters: none.

Call-back: none.

Exceptions: none.

Errors: errors will be related to various illegal parameter values or system level errors.

5.2.2 Connection open request (MauOpen)

This call is used to request the opening of the MAU-LNA connection and to send identity information to the LNA.

Pre-condition: MAU parameters defined (MAU_DEFINED).

Post-condition: waiting for LNA (MAU_OPENING) or error state.

Input parameters: optional timeout for connection attempt.

Output parameters: none.

Call-back: state changes shall be reported as described in the diagram. This includes fatally bad communication parameters.

Exceptions: possible internal resource problems.

Errors: the MAU is in an illegal state. Communication parameters are bad.

This service uses the MAPI_OREQ and MAPI_OACK messages. The result from the acknowledgement is delivered through the MAU state call-back.

5.2.3 Connection close (MauClose)

This service requests the close of the LNA-MAU session. This is done by closing the connection to the LNA and cleaning up internal state. The mechanism for closing the connection is dependent on the T-profile in use.

The closing down will take some time and the MAU will remain in the closed state until the process is completed. Note that all pending transactions and all MCP connections will be terminated without individual warning. The API is required to clean up the internal state, but the application must also do that with any state it maintains.

Pre-condition: MAU is or in the process of being connected to LNA.

Post-condition: MAU disconnected from LNA (MAU_CLOSED).

Input parameters: none

Output parameters: none.

Call-back: state change will be reported to MauState.

Exceptions: none

Errors: the MAU is in an unexpected state.

This service may use the session close message (MAPI_SESSION), but will normally just close the communication link to the LNA.

5.2.4 MAU state change call-back (MauState)

This call-back routine shall be called each time the MAU changes state. It is also called when the LNA requests a state change for the MAU or a watchdog return message.

a) EV_LNA_CLOSE: request close, can be denied by a non-zero acknowledgement.

b) EV_LNA_RESET: request restart (close and then open), can be denied by a non-zero acknowledgement.

c) EV_LNA_KILL: force close (possibly not delivered, but detected as a connection close: EV_LNA_CONNECTION_DOWN).

d) EV_LNA_STATUS: check status, used by watchdog, The API is required to close the LNA connection when the application gives a non-zero acknowledgement.

NOTE A negatively acknowledged status request will be handled as a kill, i.e. force a transition to close. A negatively acknowledged close or reset request will be handled as a status request, i.e. make the MAU remains open.

The possible events are summarized in the table below. The *return value* column specifies the return value that is expected for continuous normal operation (remaining in the open state).

Table 6 – MAU events

Event code	Normal return value	Description
EV_LNA_CLOSE	<> 0	Close request
EV_LNA_CONNECT	n/a	Connect state change
EV_LNA_CONNECTION_DOWN	n/a	Communication link down, state change
EV_LNA_KILL	n/a	Kill command
EV_LNA_NO_CONNECT	n/a	Connection failed due to timeout or missing LNA
EV_LNA_OPEN_ERROR	n/a	Port description parameter error
EV_LNA_RESET	<> 0	Reset request
EV_LNA_RESTART	n/a	Ready to re-connect state change
EV_LNA_SESSION	n/a	Remote client MAU session changes state
EV_LNA_STATUS	0	Status/watchdog request

The EV_LNA_SESSION event is passed to the MAU when a remote MAU, which previously has established a connection to one of this MAU's interfaces, dies or changes state. The session code can be retrieved by reading MCP zero's session (M_SESSION) attribute value. The state change information can be got from the last error attribute. The following session change codes are used:

a) SESSION_C0: the communication link has been restored to normal (no congestion);

b) SESSION_C1: the communication link associated with the session is congested and no more low priority messages will be accepted for transmission;

c) SESSION_C2: the communication link is also closed for normal priority messages;

d) SESSION_DOWN: the communication link to the remote MAU has been or should be closed.

NOTE 1 The session down code should normally be used to clean up internal state associated with the closed session (pending transactions mainly, but also session related structures used for example, client authentication). This may be done by calling the interface connection handlers with a "remote client dead" event. This may be handled automatically by the API.

NOTE 2 If the MAU continues to send messages on a communication link indicated as congested, this can be treated as an application error. The LNA may then close all links to the MAU and terminate the session. On the other hand, the LNA must be able to accept some more messages that may be somewhere in the communication pipeline without raising this exception.

Pre-condition: `MauInitialize` has successfully been called to define the call-back routine. The new state has normally been entered when the event is delivered (except for close requests, where denial – negative return – makes the MAU stay in the open state.

Post-condition: see above.

Input parameters: new state code and corresponding event code as defined in the figure.

Output parameters: not applicable, zero or non-zero, dependent on event.

Call-back: not applicable.

Exceptions: not applicable.

Errors: not applicable.

5.3 Session management, authentication and congestion control

A session is an association between the local MAU and one remote MAU. This means that most MCP or interface activities (see next subclause) are also related to a session (the exception is the anonymous broadcast MCPs). The local MAU cannot manipulate a session state, but the state of remote MAUs or LNAs define the state of the session.

Sessions are used for two purposes in this standard:

a) for authentication of sender of messages (connections or transactions);
b) for congestion control between two MAUs or between LNA and MAU.

5.3.1 Session control and authentication

The session states are defined by the following events:

a) the creation of the first connection between an MCP or interface on this MAU and another MAU: This defines the session. No other session-related event is associated with this;
b) any congestion related messages (`EV_LNA_SESSION`: low or normal priority congestion or no congestion) changes the "capacity" of the communication link defined by the session. The local MAU shall adjust its transmission according to these messages;
c) the loss of the connection to the remote MAU causes a session down message (`EV_LNA_SESSION`) to be delivered to the MAU. This message means that the MAU must clean up all states related to the session (close interfaces or MCPs and terminate all pending transactions). This event signals the removal of the session.

The API may decide whether the session shall be handled as an object in its own right, possibly with application related services, for example to interrogate the state. Note however, the required clean-up after a session close. Session management is handled through the `MAPI_SESSION` message and through the *session* field in the general LNA-MAU message. It is suggested that the API keeps track of the session identifier and makes this available to the application program (e.g. through the M_SESSION MCP attribute)

The LNA guarantees that the session identifier is unique to a session and that it is updated whenever an association changes (also if the remote MAU is disconnected and then reconnected).

5.3.2 Congestion control (`MauSession`)

The MAU may issue a congestion notification, either to a specific remote MAU or to the LNA (session zero). This notification should make the relevant entity stop the sending of low or normal priority messages. Congestion control does not work for urgent messages.

The MAU cannot rely completely on the congestion control notification and it should be used mostly between MAUs that have a mutual understanding or between the MAU and the LNA.

The MAU must itself cancel the congestion notification.

Pre-condition: MAU connected, and one or more sessions established.

Post-condition: session marked as congested (or unmarked) by remote MAU and/or LNA.

Input parameters: session to be manipulated, level (congested or normal)

Output parameters: none.

Call-back: none.

Exceptions: none.

Errors: bad input parameters.

This service uses the `MAPI_SESSION` message.

5.3.3 Request and connection limiting

An alternative way to limit possible congestion is to limit the number of clients or outstanding transactions on the interfaces. Interfaces can be marked with a maximum number of clients and pending transactions. These are attribute codes defined with the interface (Table 8). By setting these attributes, the MAU can limit the number of connections and/or transaction requests. The mechanism does not work for urgent requests. The limits apply to all MCPs in an interface.

5.4 Interface and connection point overview

A data object is the abstract representation in the network of connection points that are defined and served by a MAU. In both server and client MAU, these data objects are referenced by a MAU connection point (**MCP**). An MCP in an open state (with a connection between the server and one or more clients) is necessary to receive or send messages between MAUs.

The system supports two types of connection points as indicated in Figure 4:

a) the control connection point (Null MCP) that is used by the LNA for status requests, the watchdog function and MAU state control. This object is created automatically when the MAU is opened and is normally only visible to the application programmer through the close, reset, kill and status MAU events. This MCP is established automatically by the API and LNA. Transactions are handled by the API and transferred to the MAU call-back routine;

b) application defined interface MCP which is associated with an application level interface. They can only use the short data object name format, but will additionally have an interface name. These MCPs are defined in groups, one group for each interface. They are connected and disconnected in the same groups. Transactions are handled individually.

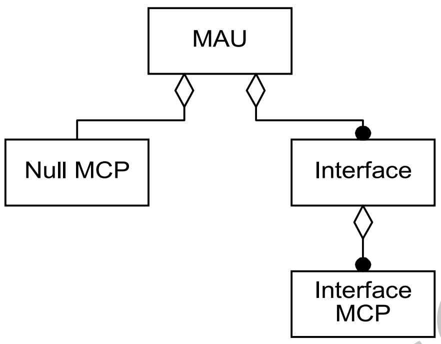

Figure 4 – Types of connection points

The control MCP does not usually need to be considered by the application programmer. The clauses following will discuss the services available for the interface MCP. The remaining part of this subclause will give an overview of MCP attributes.

5.4.1 Special considerations for anonymous broadcast (ABC)

Anonymous broadcasts are not associated to special MAUs. They are messages that can be sent to or received from any MAU. To facilitate the processing of anonymous broadcasts, the following restrictions apply:

a) an anonymous broadcast MCP can only be part of an interface called “ABCn”, where ‘n’ is replaced with one of the numbers ‘1’ to ‘5’. The interface name will determine the broadcast port used for the transmission and reception of messages. The MAU name is ignored;
b) the significant number of characters in the MCP name is 8 (the left-most characters). More characters can be used, but they will be ignored;
c) the message length is limited to 512 octets maximum;
d) the format string is hashed into a 32-bit number.

These restrictions mean (particularly item d)) that there is a possibility that differing data objects can be mixed at the receiving side. However, the probability should be low and the design tools should make sure that all ABC entries have different MCP names.

5.4.2 MCP membership in interfaces

One interface MCP will normally be associated one-to-one with an interface. However, it is possible to define “logical” MCPs that are members of two or more interfaces. This is done by creating one MCP that is registered as belonging to the different interfaces. This is useful for general MCPs that can be used in two or more different contexts. If the same MCP is used in two interfaces, it will be assigned two different MCP identity codes, one for each interface. The LNA will treat the two identity codes as distinct MCPs, but the application can use the same attribute values, including call-back functions and storage for both.

5.4.3 MCP attributes

This subclause defines all MCP attributes. The attributes are summarized in the table below.

Table 7 – MCP attributes

Attribute	M	Type	Default	Description
M_MCPNAME	F	mciname_m	*	The symbolic data object name
M_FORMAT	F	char8_m*	*	The format string
M_SEND_STORE		void*	NULL	Output data-buffer
M_RECV_STORE		void*	NULL	Input data-buffer
M_SEND_PACK		pack_m	NULL	Pack-function (marshalling for send)
M_RECV_PACK		pack_m	NULL	Unpack-function (marshalling for receive)
M_PRIORITY		int	normal	Default priority for transactions
M_TRANS_HANDLER		eventh_m	NULL	Transaction-handler
M_TRANS_RCODE		val_m	0	User return-code for transaction handler
M_LASTERROR	R	int32_m		Last error code
M_SESSION	R	word16_m		Session ID for current operation
M_INTERFACE	R	val_m		Return interface reference or NULL
M_CAN_CONFIGURE		bool_m	FALSE	TRUE if remote configurable allowed

The columns from left to right are:

– the attribute column specifies a symbolic identifier for the attribute in question;

– the 'M' column specifies how the attribute can be modified; an 'R' in this column means that the attribute is set by external events, for example a communication error, and an 'F' that the attribute cannot be changed after a connection attempt succeeded. No letter in this column means that the attribute can be changed at any time;

– the type column specifies the type of the attribute. The types are defined in 3.4;

– the default column specifies the default value the attribute gets if it is not explicitly defined. An asterisk ('*') in this column means that the attribute must be specified;

– the description column gives a short description of the attribute.

The null MCP shall support these attributes, but the interpretation of some of the values may depend on context.

Note that the remotely configurable flag is only used by special companion standards for remote configuration. It is included here for completeness and need not be implemented in all APIs.

5.4.4 Modify MCP attributes (`McpGet, McpSet`)

These two services are used to read and modify MCP attributes. This service can be called at any time and also when the MCP is open; although only attributes that are marked as connect modifiable in Table 7 can be changed when the MCP is open or opening.

Note that different variants of these routines may be necessary to modify attributes of different types, for example it may be necessary to specify a special set and get routine for function references.

NOTE For some implementations of the API, it may be convenient to code some of the attribute retrieval or modification routines as specific services associated with MCPs or interfaces. This may be particularly appropriate for object-oriented programming languages where some of the attributes can be available as object attributes or modifiable through specific methods.

Pre-condition: MCP is created(`MCP_CREATED`).

Post-condition: no change.

Input parameters: data object attribute code and optionally new value as listed in Table 7. Only those parameters listed with 'M' can be changed when the MCP is open.

Output parameters: old attribute values can be returned from the get service.

Call-back: none.

Exceptions: none.

Errors: unknown MCP or attribute code, illegal MCP state.

5.4.5 Interface attributes

This subclause defines all interface attributes. The attributes are listed in the table below.

Table 8 – Interface attributes

Attribute	S	M	Type	Default	Description
M_MAUNAME	A	F	mauname_m	NULL	The symbolic server MAU name
M_IFNAME		F	ifname_m	*	The symbolic interface name
M_IFCNAME		F	ifname_m	NULL	The name of the class of the interface
M_DESC		F	char8_m*	NULL	A description string for the interface
M_IS_ACCEPT		F	bool_m	FALSE	TRUE if accept type
M_PASSWORD		F	password_m	NULL	The interface password
M_CONN_HANDLER			eventh_m	*	Connection handler
M_CONN_RCODE			val_m	0	User return-code for connection handler
M_TIMEOUT	C	F	int32_m	0	Connection timeout (ms)
M_TRANSQUEUE	A	F	word16_m	0	Length of transaction queue
M_CLIENTS	A	F	word16_m	0	Number of clients
M_LASTERROR		R	int32_m		Last error code
M_SESSION		R	word16_m		Session ID for current operation
M_CAN_CONFIGURE			bool_m	FALSE	TRUE if remotely configurable

See 5.4.3 for a description of the columns and codes. The S column specifies if the attribute only has an interpretation for connect ('C') or accept ('A') side interfaces. For interface MCP, it is necessary to interrogate the interface attributes.

Note that the remotely configurable flag is used only by special companion standards for remote configuration. It is included here for completeness and need not be implemented in all APIs.

5.4.6 Modify interface attributes (`IfGet, IfSet`)

These services are used to read and/or modify interface attributes. This service can be called at any time, also when the interface is open.

See 5.4.4 for more information on function and possible implementations.

Pre-condition: interface is created(`IF_CREATED`).

Post-condition: no change.

Input parameters: data object attributes as listed in Table 8. Only those parameters listed with "M" can be changed when the interface is open or opening.

Output parameters: attribute values can be returned from the get service.

Call-back: none.

Exceptions: none.

Errors: illegal attribute codes or values, MCP in wrong state.

5.5 Interface management

The state diagram for an interface is shown below.

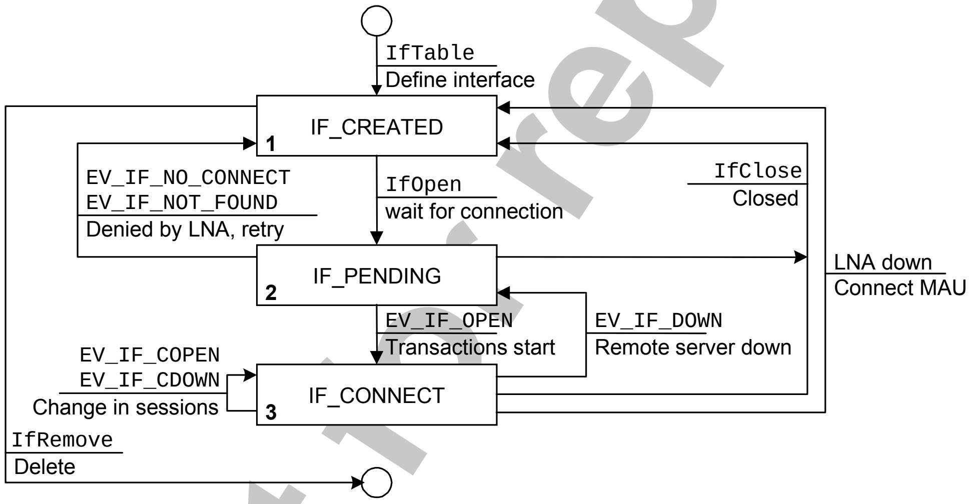

Figure 5 – Interface state transitions

The states are:

a) `IF_CREATED`: the interface has been defined by the application, but is not yet connected. The definition can be done while the MAU is in the created state (without connection to the LNA). Note also that if the LNA connection goes down, all interfaces will automatically be set to this state;

b) `IF_PENDING`: the interface attributes have been sent to the LNA. The MAU is waiting for acceptance. There is a difference here between accept and connect objects:

   1) accept object attributes are only exported to the local LNA. The object will only stay in this state until the local LNA accepts or denies the registration. This will normally only happen if there is an identical interface or MCP already defined by this MAU. A connection timeout has limited utility,

   2) connect interface attributes are exported to the system for connection to a remote MAU's accept interface. The interface will stay in this state until the remote MAU is found and has accepted or denied the connection or until the timeout triggers;

c) `IF_CONNECT`: the interface has been registered and it is possible to receive or initiate transactions.

States b) and c) are only used when the MAU is in the open state. The closing of the MAU will transfer all interfaces back to created. The API is required to do necessary state clean up and terminate all pending transactions. The application is required to do a similar clean-up in its data structures.

This standard requires that all MCPs in a requested interface are accepted on the server side for the connection attempt to be positively acknowledged. This means that the client side interface must be a true subset of the accept side interface. Errors of any type shall be reported to the application.

If the MAU enters the error state, no call-backs will be delivered, neither on pending transactions nor closed connections. The interfaces will be in an undefined state.

The reconnection of a broken interface connection (from state c) to b) and back to c)) will be done automatically by the LNA. Note that the timeout, if specified, will bring the interface to state a), also after an automatic LNA reconnect.

The following subclauses define the services that are available for interface management.

5.5.1 Define interface (`IfTable`)

This service is used to define MCPs and MCP attributes for later export to the LNA. It defines a new interface and sets it to the created state. It can in principle be called at any time (see "M" marked attributes in clause 5.4.5), but it is most useful for initial creation. It specifies a set of attributes for the newly created data object.

Pre-condition: MAU is created(`MAU_DEFINED`).

Post-condition: `IF_CREATED`.

Input parameters: the attributes are partly common attributes to all MCPs in the interface and partly attributes associated with each MCP. It is suggested that the input format is in the form of a table that lists interface attributes, MCP attributes and attribute values in a format that can be automatically generated by a computer tool.

Legal interface attributes and attribute value types are those listed in Table 8. MCP attributes are listed in Table 7. The MCP attributes marked with 'F' cannot be set in an interface definition. Attributes in both tables marked with an asterisk must be defined before the interface can be opened.

Output parameters: interface reference.

Call-back: none.

Exceptions: none.

Errors: illegal MAU state, illegal interface state, illegal attributes or attribute values.

5.5.2 Remove interface (`IfRemove`)

This service is used to remove an interface, its MCPs and all associated state. It is only legal to do this when the interface is in the created state.

Pre-condition: `IF_CREATED`.

Post-condition: interface deleted.

Input parameters: interface reference.

Output parameters: none.

Call-back: none.

Exceptions: none.

Errors: illegal interface reference or state.

5.5.3 Establish interface connection (`IfOpen`)

This service exports the definition of an interface to the LNA. It shall send the relevant messages to the LNA and wait for its reply. Two cases must be considered:

a) an accept type interface will cause the creation of network data objects and is handled by the local LNA alone. There are few restrictions on how the interface and the MCPs are constructed;
b) a connect type interface shall be connected to a set of network data objects and needs to match the intended targets in all connection oriented attributes (names and format strings). For the connection attempt to succeed the following pre-requisites exist:
    1) the connection related interface attributes must match on both sides,
    2) the connection related attributes in all the client defined MCPs must match (a subset of) the server defined MCPs. The client does not need to connect to the whole server interface, but all MCPs in the client interface must connect to a corresponding server MCP for the connection to succeed.

Pre-condition: interface is created(`IF_CREATED`) and all necessary attributes are defined.

Post-condition: `IF_PENDING` and then possibly `IF_CONNECT`

Input parameters: all connection-related attributes from the definition of the interface and its MCPs are used in executing this service. A time-out different from zero will make the LNA give up its connection attempt when the specified time has expired. Zero will make the LNA keep on trying to reconnect. The time-out has no function for accept type interfaces.

Output parameters: none.

Call-back: the interface connection handler is called at all further interface state changes. Transaction handlers are defined for each MCP. These will be called on incoming transactions.

Exceptions: timeout and format errors. The LNA may give other interface related error messages on some occasions. All exceptions will be delivered to the connection handler.

Errors: illegal interface state, illegal or undefined attribute values, illegal interface reference.

This service uses the `MAPI_IREQ` and `MAPI_IACK` messages. The result from the acknowledgement is passed back through the interface connection call back routine.

5.5.4 Close interface (`IfClose`)

This service removes an interface from the connected state. It can be used in the pending or the open state. The service will block any further transactions on this object. This service will signal to all remotely connected MAUs (server or clients) that the interface is down.

Pre-condition: interface is pending or open(IF_PENDING, IF_CONNECT).

Post-condition: IF_CREATED.

Input parameters: reference to the interface.

Output parameters: none.

Call-back: none.

Exceptions: none.

Errors: illegal interface state, illegal interface reference.

This service uses the MAPI_IREM message.

5.5.5 Interface state change call-back (InterfaceState)

All state changes for the interface that are not an immediate effect of a service call will be reported to the interface connection call-back routine. In addition, it will be called when a remote MAU tries to connect to the interface.

Pre-condition: interface is pending or open(IF_PENDING, IF_CONNECT).

Post-condition: state has changed immediately before call.

Input parameters: event code and interface reference, user defined reference (value of attribute M_CONN_RCODE). All event codes are listed in the table below.

Table 9 – Interface events

Event code	Return value	Description
EV_IF_OPEN	n/a	Interface open
EV_IF_DOWN	n/a	Interface closed from remote side
EV_IF_COPEN	0	Client requests connection to accept interface
EV_IF_CDOWN	n/a	Client disconnects from accept interface
EV_IF_NO_CONNECT	n/a	Attribute values inhibit the establishment of the interface
EV_IF_NOT_FOUND	n/a	Connection timed out or was denied due to, for example high load

NOTE The EV_IF_CDOWN event is only delivered (as an A-profile message) when the remote client closes down a connection in a controlled manner. If the remote MAU dies, this will be notified through a session close event to the MAU (see below).

The *no connect* code is used when attribute values inhibit the connection attempt, for example password errors, badly formatted names, etc. The *not found* code is used when the attempt was aborted because the data object was not found (timeout) or because connection was temporarily denied. Details about the cause can be by examining the *last error* attribute value for the interface.

The client connection attempt can be denied by the server for any reason. The normal procedure will be to have a separate authentication interface (see the companion standards) that is used to authenticate operator and/or physical host computer on the client side. By comparing the session identifier (M_SESSION attribute) for the new interface connection attempt and the previous authentication transaction, the identity of the client can be established and used to deny or accept the connection attempt.

The death of the local LNA should cause all interfaces to go to the created state, possibly without call-back to the connection handlers. The call-back will be dependent on the API implementation. The death of a remote client MAU will only be reported to the MAU state change handler (see 5.2.4) as a session close event. If necessary, the application should use the session down event to reset the state related to remote clients. The handling of these cases may vary between APIs.

Output parameters: client connection attempts can be denied by returning a non-zero value.

Call-back: not applicable.

Exceptions: not applicable.

Errors: not applicable.

5.6 MCP transactions.

Transactions look somewhat different for the different types of MCPs and whether it is seen from a client or a server. This subclause will be further sub-divided to define the functionality and semantics of each of the main group of transactions. Table 10 summarizes the different transaction types. Transactions will be described for both client and server type MCPs in each subclause. Differences will be noted where appropriate.

Table 10 – Transaction types

Transaction type	Client sends	Server sends	Server later initiates
Function	Data W	Data R	
Read	Request	Data R	
Write	Data W	Acknowledge	
Non-acknowledged write	Data W		
Subscribe	Request	Data R	Data R to all clients
Broadcast subscribe	Request	Data R	Data R to all (unreliable)
ABC subscribe		Data R	Data R to all (unreliable)
Individual subscribe	Data W	Data R	Data R individually

All transactions are associated with one or two data structures: data W and data R. Data W is sent by the client, data R is sent by the server, usually as an effect of the arrival of data W. This is typical of the basic function type transaction. The read transaction has an empty data W from the client (it is a request for data only). The write type transaction contains data from the client to the server and a per transaction acknowledgement, without data, from the server. To save bandwidth, the non-acknowledged write does not allow the server to send an acknowledgement. This group of transactions terminates when the last message has been processed by the receiver.

For subscribe type transactions, the server may send more data R after the first acknowledgement. The subscribe transactions do not terminate until explicitly cancelled, either by a cancel request or by the death of the client or server. Note that the individual subscribe allows the client to send data to the server. This data can be used to configure different replies (also for server initiated transmissions) for different clients. Other subscribe forms send the same data structure with the same contents to all listening clients. Anonymous broadcast (ABC) subscribe does not use the normal client/server paradigm. It allows any node to send to or listen on any other node, without knowing the identity of either.

Each transaction (except ABC subscribe) is initiated by the client by sending a message, possibly with some application specific information to the server. The message is handled by the server through its transaction call-back routine which may generate an acknowledgement to the client. The acknowledgement is then finally processed by the client in its transaction call-back routine. Different variants of this pattern will be used for the different types of transactions.

These services uses the messages described in 8.2.4.

5.6.1 Function type transactions

These transactions are of the following type:

a) function call with application information from client to server and back again;

1 read request with application information only from server to client;

1 write with application information only from client to server, but with acknowledgement from server.

The state diagrams for the client (left) and server (right) side transaction are shown in Figure 6.

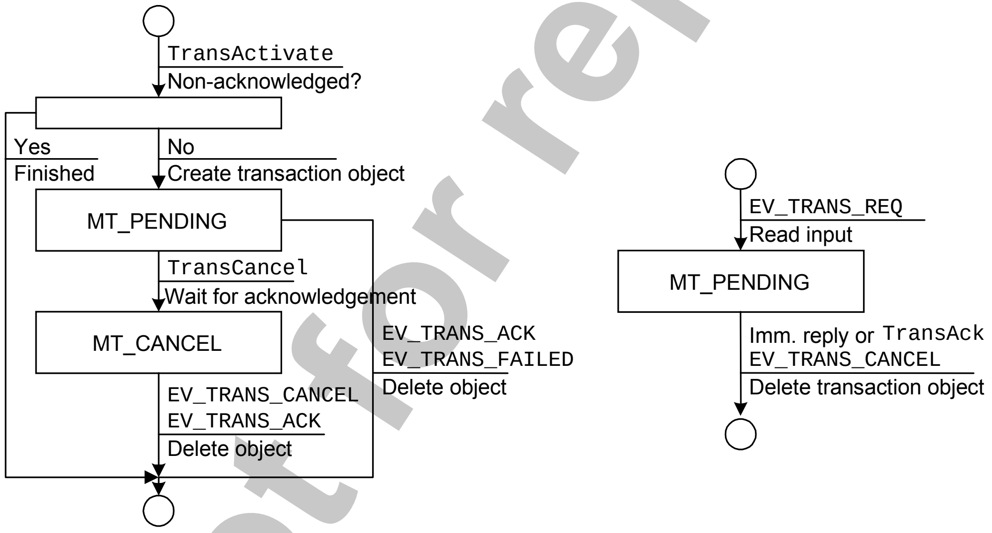

Figure 6 – Function type transaction states

On the client side, the object is created when the client initiates a transaction through a call to `McpActivate`. This causes the request and a write data structure to be sent to the server. When received by the server, the request will be delivered to the server's transaction handler and instantiated as a transaction object. The server may send an acknowledgement immediately by signalling this on return from the handler or delay the acknowledgement until later. For later acknowledgements, the server must issue a call to `McpAck`. For late acknowledgement, the server's transaction handler may be called if the client cancels the transaction. The cancellation can also be caused by a timeout. In both cases, an `EV_TRANS_CANCEL` event is delivered.

The client may cancel a pending transaction by calling `McpCancel`. The client may also specify a timeout that can be used by the LNA to cancel the request. Both events causes a cancellation message to be sent to the server. In the timeout case, the notification will be delivered to the client as an `EV_TRANS_FAILED` event.

The cancellation will take effect for the client as soon as it has been processed by the client's LNA and returned as a cancellation acknowledge (EV_TRANS_CANCEL). However, the cancellation may come after the request has been returned by the server (EV_TRANS_ACK), in which case a negative cancellation acknowledge is delivered to the client, after the data acknowledge (see sequence diagrams). The negative acknowledge will then refer to no transaction object and should normally be ignored by the API or the application.

5.6.2 Non-acknowledged write transactions

This transaction can be seen as a one-way write. No acknowledgement is delivered from the server. The transaction is also stateless. It consists of a message that leaves no trace and it cannot be timed out or cancelled.

Note that delivery is reliable as long as the session between client and server is intact.

5.6.3 Subscribe type transactions

These transactions are of the following types:

a) normal subscription where the server sends one R-data message to the local LNA which copies the message to all clients via reliable links. On initial subscription, the server will send one R-data message to the subscribing client alone;

b) broadcast subscribe which is identical to the previous, except that the LNA repeats all but the initial R-data message via an unreliable broadcast link. The initial exchange goes as for a normal subscription;

c) anonymous broadcast (ABC) where sender is unknown to receiver. In this type of transaction, there is no initial request and acknowledgement.

The state diagrams for the client (left) and server (right) side transaction are shown in Figure 7. Note that the server side diagrams are divided in two: A general object (upper) that applies for all subscription types and an initial request transaction object (lower) that only applies to normal subscriptions (not broadcast).

The server side does not usually see the identity of the clients. All operations are performed independently of what or how many clients there are. This means that the server side most of the time needs a single transaction object that conceptually is created when the MCP is connected. Per client transactions are received as the initial request (EV_TRANS_REQ) only and only on normal subscriptions (not broadcast types). The server must reply to these requests as for a normal read request. This means that the acknowledgement can be delayed. When the first request is acknowledged, the client MAU is added to the subscriber list by the LNA and all R-data messages are sent to all clients as a result of TransAck calls. The LNA takes care of distributing the data messages as necessary. Note also that the server has no possibility to check if some clients have cancelled their connection. This is handled by the LNA without involving the server MAU.

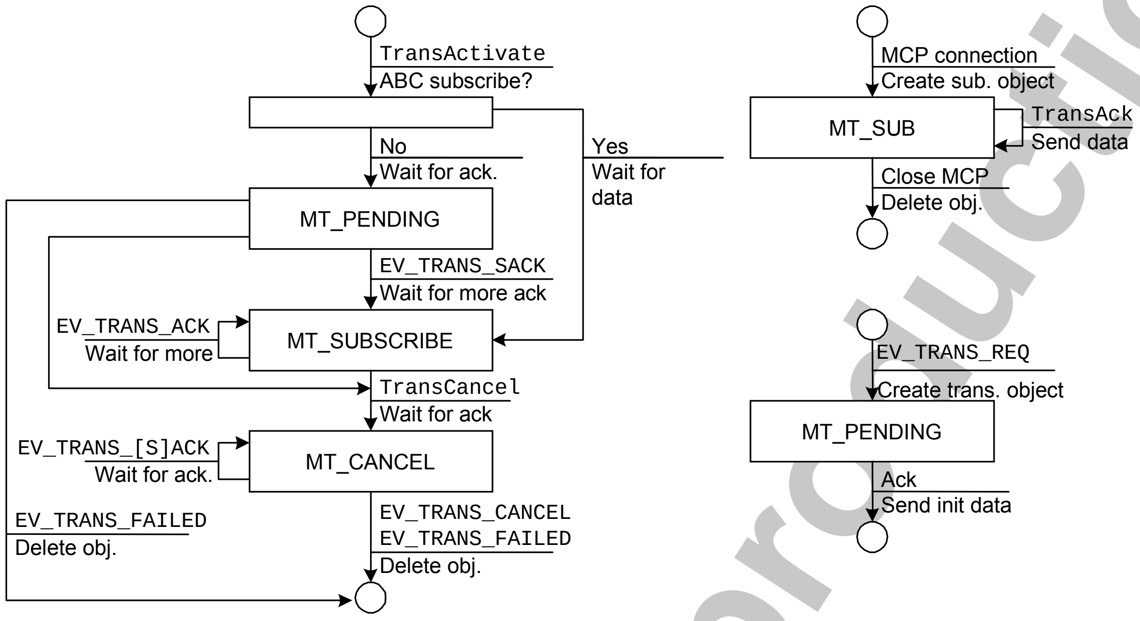

Figure 7 – Ordinary subscribe transactions

The client initiates the transmission of R-data from the server by using a TransActivate call. The result of this call is that one individual R-data message is received (EV_TRANS_SACK – as the result of the EV_TRANS_REQ event on the server side). Later, only collective messages are received (EV_TRANS_ACK) as results of the server's TransAck activations. The client can cancel the subscription by using the normal cancellation service. One or more data messages may still arrive during the period the client waits for the cancellation to take effect. Note also that a timeout, in principle, can terminate a subscription attempt. However, as the server answers immediately to a subscription request, this should normally be a rare occurrence.

5.6.4 Individual subscribe transactions

Individual subscriptions can be looked at as a variant of the normal function type transactions. They let the server maintain a list of client specific transaction objects and answer to these individually. The state transaction diagrams are shown in Figure 8, with the client at the left side and the server at the right.

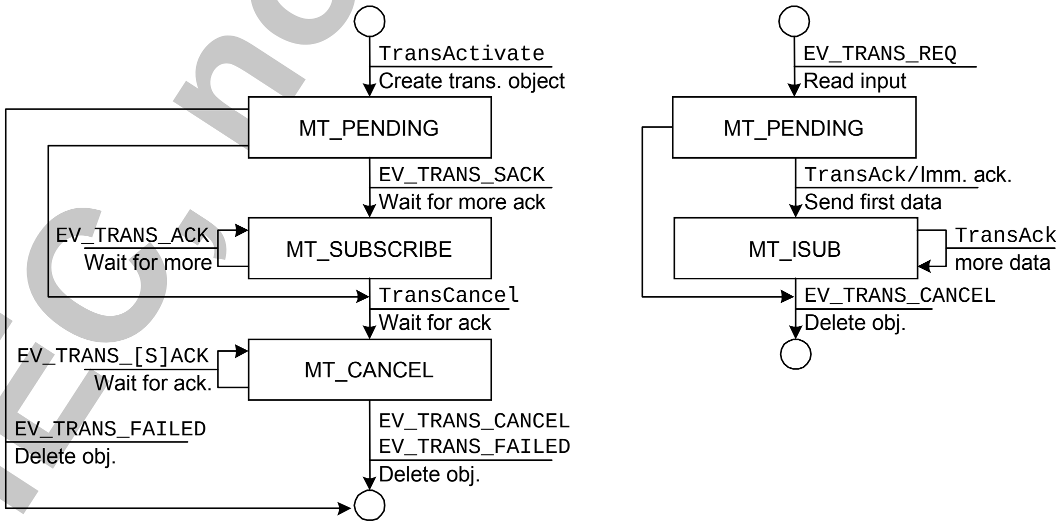

Figure 8 – Individual subscribe transaction states

The difference from the normal subscription is that the server side treats each transaction as a separate object and can send data to each client individually. The state on the server side differs from the function type transaction objects in that they in addition to the delayed acknowledgement state (`MT_PENDING`) need a state for subscription in progress (`MT_ISUB`). Note that the transition from pending to subscribed either happens when an immediate acknowledgement is given or at a later stage, when a delayed acknowledgement is sent. On the client side, the states are identical to the normal subscription states.

5.6.5 Client side initiation (`TransActivate`)

This service creates a transaction object on the client side. Any outgoing data (W data) must be ready before the service is called. It is possible to specify a time-out for the transaction. It is also legal to cancel the transaction.

Note that time-out and cancellations may apply to the client side. A timed out or cancelled transaction will often have been processed by the server and an acknowledgement sent. In this case, the server will normally not get the cancellation message.

Pre-condition: MCP is connected (`MCP_OPEN`). W data is ready if applicable.

Post-condition: new transaction object created.

Input parameters: W data if applicable (normally via the output store MCP attribute).

Output parameters: reference to transaction object.

Call-back: return value, cancellations or timeout.

Exceptions: see call-back.

Errors: bad parameters.

5.6.6 Transaction cancellation (`TransCancel`).

This service cancels a transaction from the client side.

Pre-condition: transaction object exists.

Post-condition: waiting for acknowledge.

Input parameters: reference to transaction.

Output parameters: none.

Call-back: cancellation. Note that return value(s) may arrive before cancellation acknowledge.

Exceptions: none.

Errors: bad input parameters.

This service uses the `MAPI_CREQ` and `MAPI_CACK` messages. The result is reported back to the transaction call-back routine.

5.6.7 Client side transaction state change (`TransClientState`)

This service notifies the client of the completion, cancellation or time-out of a transaction. The service will get access to any incoming data (R-data).

The client's transaction handler is normally defined as one of the MCP attributes. However, a specific API implementation may, for example define the transaction handler when the transaction is initiated.

Pre-condition: existing transaction object.

Post-condition: transaction object deleted, except for subscription objects.

Input parameters: incoming R data, if any. Event code as listed in the table below.

Table 11 – Client transaction events

Event code	Return value	Description
`EV_TRANS_ACK`	n/a	Transaction completed (late subscription R-data)
`EV_TRANS_SACK`	n/a	First subscription R-data
`EV_TRANS_CANCEL`	n/a	Transaction successfully cancelled
`EV_TRANS_FAILED`	n/a	Transaction failed (timeout)

Output parameters: none.

Call-back: not applicable.

Exceptions: not applicable.

Errors: not applicable.

5.6.8 Server side transaction state change (`TransServerState`)

NOTE The server's transaction handler must be defined as one of the MCP attributes. Client transaction requests will arrive independently of the activities of the server.

This service notifies the server of an incoming transaction. The service will get access to any incoming data (W data) and must prepare any outgoing data (R data) before terminating.

Pre-condition: MCP is connected (`MCP_OPEN`) and a client has connected.

Post-condition: dependent on transaction type, see state diagrams.

Input parameters: incoming W data, if any. Event code as listed in the table below.

Table 12 – Server transaction events

Event code	Return value	Description
`EV_TRANS_REQ`	n/a	Transaction request from client
`EV_TRANS_CANCEL`	n/a	Transaction cancelled by client

Output parameters: outgoing R-data, if any. Flag to send outgoing data when appropriate. This operation is handled by the API and is API dependent.

Call-back: none.

Exceptions: none.

Errors: none.

5.7 Bulk data transfer

Bulk data transfer is a mechanism whereby large amounts of unformatted data can be transferred in a reliable and sequential stream. Bulk data transfers are performed via special purpose data sinks or sources that are provided by the T-profile directly.

To use bulk data transfer, the MAU has to have access to the same T-profile as that used by the LNA. This can in principle be achieved independently of the conformance class of the MAU [IEC 61162-400]. However, the T-profile must support the priority mechanism used by the LNA and should also support the same level of redundancy.

The bulk transfer links are established directly through the T-profile. However, the companion standard will define a bulk data transfer address exchange interface. The necessary prerequisites are a common convention for the specification of network addresses. This is described in the T-profile. The API will normally provide services for establishment of bulk source and sink ports.

6 Functional requirements for LNAs

This clause specifies the basic functionality requirements for the LNA. It is mainly written to assist the designers of the LNA, but it also contains normative descriptions of various relationships between protocol entities.

6.1 Context diagram and functional overview

The context diagram for the LNA is shown in Figure 9.

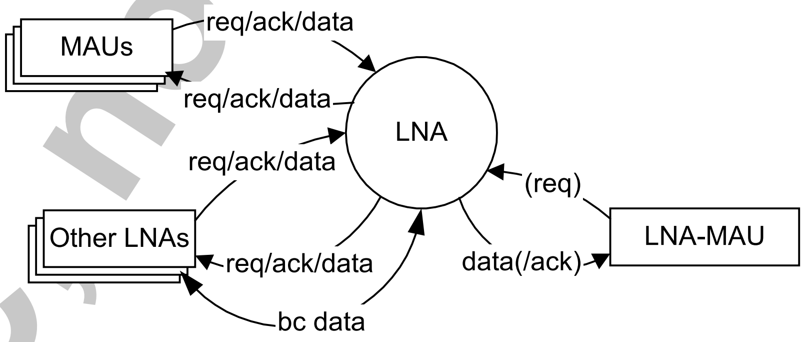

Figure 9 – Context diagram for LNA

The LNA's main task is to act as a multiplexer for messages from MAUs and demultiplexer for messages to MAUs. All MAU-to-MAU traffic goes via one or two LNAs. One LNA is used if the MAUs are clients of the same LNA, otherwise two LNAs are used. To work as a message handler, the LNA also has to take care of session management, both with regards to MAUs and to MCPs.

The LNA must also provide name look-up functionality. The data objects that are used for connection establishment are only identified with various textual attributes. The LNA must provide the functionality to map data object attributes to network level addresses (of other LNAs) and MCPs on client and server MAUs.

The LNA's tasks can be summarized as

– MAU name look-up:
  – maintain a database of the MAUs that are served by the local LNA. Respond to MAU name interrogations from other LNAs,
  – maintain a database of remote MAUs and their location. Perform name look-up, i.e. broadcast requests when a required remote MAU is not already in the database,
  – manage remote LNA sessions (connect to and disconnect from remote LNAs);
– MAU session control:
  – manage MAU sessions. Let MAUs connect, change their state and disconnect,
  – manage data object connections. This has two aspects: export local data object definitions to the network and connect local client MAUs to remote data objects;
– Message multiplexing and de-multiplexing:
  – manage transactions. The LNA services all transactions that relate to local MAUs, either as clients or servers.
– Other services:
  – give diagnostic services through the LNA MAU.

MAU name look-up is based on unreliable broadcast. None of the transmitted information is guaranteed to reach all or any remote LNAs. All functionality in the LNA name look-up part is based on re-transmitting requests and responses until the system state has stabilized. This means that the MAU name information exchange must be modelled as a process where messages can arrive at any time.

Other traffic between MAUs, remote LNAs and the local LNA is generally of two types:

a) request/acknowledge pairs involving the local LNA, or
1 data transport messages that are just routed through the LNA. Both are normally sent on reliable point to point links. The exception is broadcast subscription messages that are sent on an unreliable broadcast link.

One LNA shall be able to service two local MAUs that are communicating. This standard will assume that this type of communication uses the normal LNA-LNA message types and is performed through a loop-back communication link implemented internally in the LNA. Although convenient for the designer, this is not mandatory. Other solutions conforming with the standard can be envisaged.

The functionality of the LNA MAU is described in the companion standard. It shall be able to provide an overview of the application level network of MAUs, of traffic related statistics and various other parameters. It shall have the functionality of any normal MAU, but shall be given a unique name as described in 8.4.3.5.

6.2 MAU name management (MauAck, MauRequest, SessionClose)

The LNA is required to maintain an internal database of MAUs known to it. This includes local as well as remote MAUs. The database shall also be made available to the LNA MAU. Associated with this database shall be a MAU name function that provides the following services:

a) provide look-up functionality for local MAUs that search for a remote MCP server. Each request (MauRequest) shall have an individual overall timeout and shall be acknowledged (MauAck) either at timeout or when the MAU was found. These requests and acknowledgements are provided and used by other functional components of the LNA;

1 provide MAU connection management service, establishing the MAU level connection with LL_MAUREQ and LL_MAUACK messages;

1 a session close reporting function (SessionClose) that informs all other functions in the LNA about the closing down of a remote or local LNA;

1 a general reporting service to other LNAs about which MAUs this LNA serves or has ceased to serve. Local MAU state is reported from the MAU management function with NewMau. General broadcast information is sent via CC_DEFMAU, CC_DEADMAU and/or CC_WATCHDOG. Directed information is sent to all connected LNAs with LL_MAUACK(N) messages or to MAU-requesting LNAs with LL_MAUACK messages.

Figure 10 shows the state transition diagram for MAU definitions in the database. Each MAU is represented with exactly one state diagram (see note below). The state MAU_LDEFINED represents a defined local MAU and the state MAU_DEFINED represents a defined remote MAU. The state MAU_UNDEFINED represents pending requests for a (possibly remote or not yet connected local) MAU. The event labels typeset in courier represent incoming messages to the LNA. The courier action labels represent outgoing messages.

Other states are MAU_FOUND for MAUs that have been reported to exist on an LNA, but has not yet been confirmed. MAU_HEARD is an optional state which can be used to store information from name broadcasts that is not immediately useful.

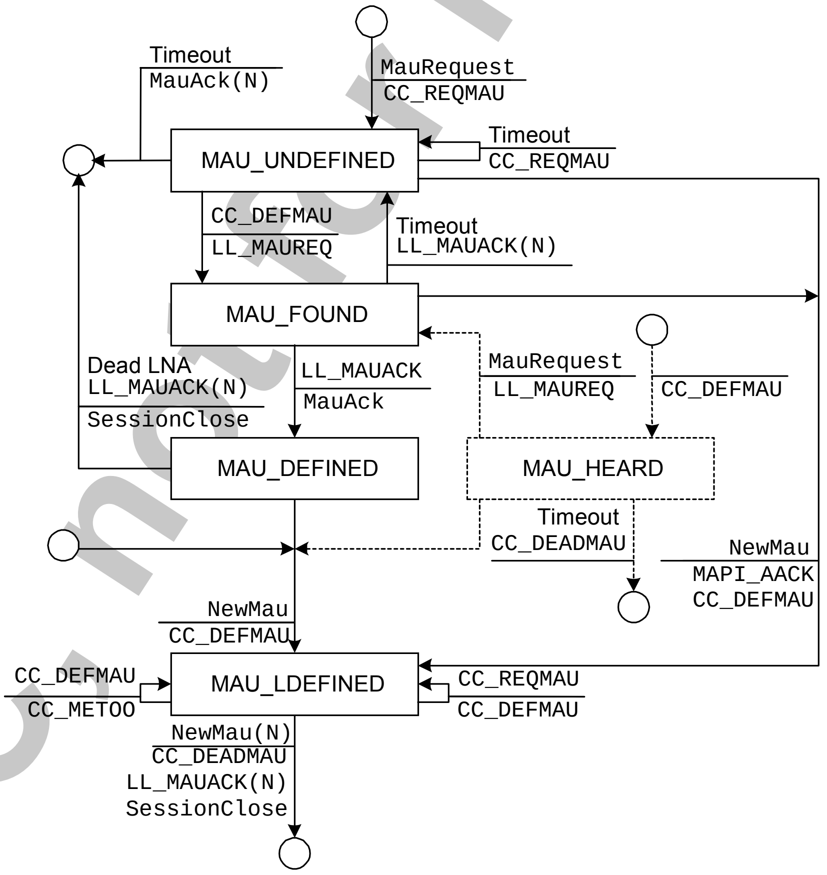

Figure 10 – LNA's MAU database state definitions

NOTE If more than one local MAU requests a connection to the same remote MAU (via a sequence of calls to MauRequest), only the first local MAU's call should lead to an exchange of MAU request/acknowledge messages. For the following MAUs one can immediately start on the interface connection attempt or wait until the first MAU's connection attempt succeeds or fails.

6.2.1 Duplicate MAU names

The LNA will check its database for duplicate MAU names, but it will not inhibit the creation of MAUs with names that are already registered as long as no more than one of them is local. In the case of two local MAUs with the same name, the latest to register will be denied. In the case of duplicates with one of them being remote, this shall be signalled by broadcasting a `CC_METOO` message. It shall also update its database with the name pointing to the latest local MAU identity.

Note that duplicate MAU names, even when handled on two different LNAs cause problems when establishing new connections to the MAU. The MAU that the connection is established to will depend upon which of the MAUs is first selected by the connecting LNA. This will, in general, be arbitrary. Existing connection will not be influenced by the addition of a new MAU with the same name. However, once the old MAU dies or disconnects, all connections are immediately moved over to the newer MAU. This mechanism can in principle be used to create functional redundancy.

6.2.2 Local MAU names

Local MAUs are added to the database in the `MAU_LDEFINED` state when they connect to the LNA (`NewMau`). The LNA shall immediately send out a name broadcast. The same applies when a local MAU closes and is removed from the database. A broadcast shall immediately be sent, together with a reliable `LL_MAUACK(N)` message to other connected LNAs. The LNA shall inform all other LNAs that maintained sessions with the dead MAU (through local MAUs that acted as servers or clients to the dead MAU). It is allowed to send the information to all remote LNAs. The LNA must at all times maintain all known and valid local MAU names in its database. The information to other LNAs shall also be sent when remote LNAs request information about MAUs in the database.

Relevant messages for name reporting are:

– `CC_DEFMAU`: used to inform remote LNAs about a new MAU;
– `CC_METOO`: used to inform remote LNAs that a duplicate MAU name has been detected;
– `CC_DEADMAU`: used to inform other LNAs that a local MAU has been removed;
– `CC_REQMAU`: used by remote LNA to query this LNA about a MAU location;
– `CC_WATCHDOG`: similar function as `DEFMAU` and `REQMAU`, but with somewhat different semantics (see 6.3.4);
– `LL_MAUACK(N)`: used by local LNA to notify other LNAs about the death of a local MAU (session close).

The messages used to verify and deny MAU location at a given MAU are used in conjunction with interface or MCP connection establishment. In addition to the state transition diagram shown above, the reader is referred to the appropriate sequence diagrams.

6.2.3 Remote MAU names

The state diagram shown in Figure 10 also applies to the identification of remote MAUs. The LNA needs to maintain a database of remote MAU names and locations, i.e. the physical network address of the MAU's LNA. This database shall, as a minimum, contain all the remote MAUs connected to by local MAUs (all sessions, see 5.3). The database can also contain information about "heard about" MAUs, these are put into the MAU_HEARD state. This state can be used to speed up connection attempts by using information already being transmitted in the system. If this state is used, the LNA should have a kind of time-out mechanism to remove old and expired MAU names from this state. Reported dead MAUs must also be removed.

Each time one of the LNA's MAUs requests a connection to a remote MAU, the LNA will search in the database for the location of the remote MAU. If the location is not found in the database, the LNA shall flag the MAU as undefined (MAU_UNDEFINED) and broadcast a MAU request message to the other LNAs. Other LNAs will send solicited (as a response to the above name query) and unsolicited information about the status of their MAUs (CC_DEFMAU). Any match should move the requested MAU to the MAU_FOUND state.

The LNA shall send periodic requests for undefined MAUs and it shall continue to do this until all pending connection attempts terminate, either by the MAU being found or by connection request timeout. The periodic requests should preferably be sent in CC_WATCHDOG messages, but they can also be sent in CC_REQMAU messages. The period for transmission shall be increased to avoid broadcast storms during system initialization. The transmission intervals should be as follows:

- first request shall be sent immediately;
- second request after 1 s (RETRY_INTERVAL);
- third and fourth request with intervals of twice the retry interval after the second attempt;
- fifth and sixth requests with intervals of three times the retry interval after the fourth attempt;
- further requests 5 s to 10 s (1 to 2 times WATCHDOG_INTERVAL) apart, depending on the number of missing MAUs and synchronization with the watchdog function.

In addition to these intervals, each individual MAU request will have its own timeout interval. Acknowledgements, positive or negative, shall be given at this time, no later. The actual semantics of this function are dependent on the functionality of the LNA and are closely linked to the function of connect MCP and interface management.

The overall MAU interrogation interval is set so that it shall be possible to synchronize the requests to the transmission of the CC_WATCHDOG message.

When a remote LNA reports the missing MAU name, the MAU is put in the MAU_FOUND state. In this state, the local LNA checks the information by interrogating the remote LNA. This is done by sending the LL_MAUREQ request. The answer shall be a positive or negative LL_MAUACK. If the acknowledge is negative (no such MAU exists) or if the connection to the LNA goes down, the MAU will be reset to the MAU_UNDEFINED state and the request process continued. The retry intervals may or may not be reset to their previous state. If the acknowledgement is positive, the MAU is moved to the MAU_DEFINED state, and the connection attempt can continue.

The MAU is removed from the MAU_DEFINED state if the connection to its LNA fails or if the LNA reports the MAU dead. It shall also be removed if no more connections to the MAU exist and it is reported dead by a CC_DEADMAU message. The MAU definition is also moved from the defined state if a local MAU connects with the same name. In this case, the new state is locally defined, but the connections to the remote MAU are kept.

The following data messages from other LNAs will indicate invalid state information in the remote MAU database:

– LL_MAUACK (negative). Used by remote LNA to report that the requested MAU was not found or that it died.

The death of an LNA shall also be interpreted as the death of all MAUs that the LNA serves.

The following name look-up messages (BC) are used between LNAs:

– CC_REQMAU: used by local LNA to query other LNAs about a remote MAU location;
– CC_DEFMAU: used by remote LNAs to inform about a new or requested MAU;

– CC_METOO: used by remote LNAs that have a duplicate MAU name in their database;
– CC_DEADMAU: used by remote LNAs to signal that a remote MAU has been removed;
– CC_WATCHDOG: alternative to all above except me too message.

The LNA shall, if possible, report all received messages about duplicate MAUs, for example on a diagnostic console or to the LNA MAU.

All incoming requests for remote MAUs shall be checked against all MAU definitions in any state in the data base. This is not shown explicitly in the state transition diagram. It is important to design the LNA so that consistency in the data base is kept, particularly between locally and remotely defined MAUs.

6.3 LNA-LNA session management

The LNA is required to periodically check the liveness of all connected LNAs. This is done by broadcasting a status report and listening to other LNAs' status reports. The LNA is also required to maintain an up-to-date list of known LNAs. This list shall be made available to the LNA MAU. The below diagram describes the known LNA's state changes.

Note that the dead state is optional in that the LNA can remove disconnected LNAs directly rather than keep them as dead. It is, however, useful for system management to keep a record of known, but dead LNAs. The list of dead LNAs, if used, should be purged, for example by using a timeout for old definitions.

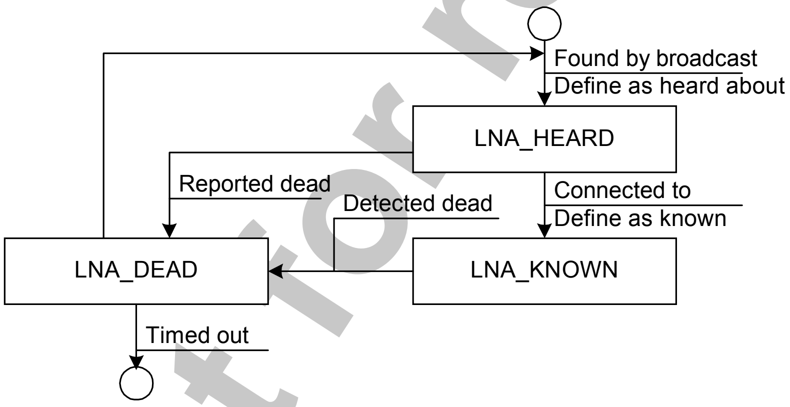

Figure 11 – LNA states

See also 6.3.4 for more information about the LNA-LNA watchdog function.

6.3.1 Heard about LNA

The broadcast messages that can report an LNA are:

a) CC_DEFMAU: a remote LNA reports its MAU;
b) CC_WATCHDOG: general watchdog messages sent by other LNAs.

6.3.2 Known LNA

A remote LNA shall only be defined as known when the communication link arbitration phase is completed. This is handled by the T-profile. This phase shall ensure that only one link is established between the two LNAs. Arbitration is necessary when two LNAs are connecting to each other simultaneously. The arbitration is done by the T-profile.

6.3.3 Dead LNA

The events that can be used to detect a dead LNA are the following:

a) the connection from this LNA to a remote LNA is detected dead by T-profile functionality. See below for more details;

1 CC_WATCHDOG: a message sent from other LNAs reports a “heard about” LNA dead;

1 CC_WATCHDOG: the lack of an expected watchdog message from a “heard about” LNA defines it as dead;

1 LL_ALIVE: not responded to after an item c) watchdog miss as described below.

Items b) and c) only apply to the transition from “heard about” to dead. They shall not be used to define a connected (known) LNA as dead.

6.3.3.1 Connection loss

The connection shall be considered dead only if the urgent and/or the normal priority link closes. The LNA may try to continue operation if only the low priority link dies. Congestion shall not be considered a lost connection.

6.3.3.2 Watchdog trigger

Item c) shall, if the LNA was connected, be followed up with a reliable link check before the LNA is defined as dead. The check shall be done by sending a liveness enquiry (LL_ALIVE) and wait for one more watchdog interval to see if an answer or other traffic arrives (including CC_WATCHDOG). In case of normal priority link congestion, the waiting time shall be doubled. If the sending of the LL_ALIVE causes a link failure, the remote LNA shall immediately be defined as dead.

6.3.4 The LNA watchdog function

The LNA is required to maintain a watchdog function to check that all connected LNAs are alive. This function is based on periodic “is alive” broadcast messages from all LNAs. A failure to receive a timely broadcast shall make the LNA check the communication link by sending a reliable liveness request (LL_ALIVE). The watchdog broadcast shall also contain information about topology in the system and, if space permits, information about local and undefined MAUs. The message is called CC_WATCHDOG.

The message shall be sent with a minimum of 5 s intervals (WATCHDOG_INTERVAL). It should normally not be sent more frequently, except as necessary to report undefined MAUs. The message shall always contain the local LNA identity. Following that, it shall contain undefined MAUs with a request frequency as defined in 6.2.3. If necessary, more than one message may be sent at each watchdog interval, each containing one distinct set of undefined MAUs. Note also that the longest retry interval is double the normal watchdog interval which allows two consecutive messages at the normal watchdog interval to contain different sets of MAUs. If there is space, the watchdog message may also contain a list of local MAUs. This list should be created so as to ensure that all local MAUs are reported eventually.

The watchdog message shall be used to monitor the liveness of remote LNAs. A remote LNA which has not sent a watchdog message within the specified interval shall be checked for liveness by checking the reliable links to the LNA (LL_ALIVE).

6.4 Local MAU connection management

6.4.1 MAU identification

6.4.1.1 MAU name

MAUs shall be uniquely identifiable by their MAU name. An LNA shall deny services to a MAU with the same name as another local MAU. If two remote MAUs have the same name, the second arrival shall be discarded from the LNA's database after a warning to the reporting MAU (see 6.2.1).

6.4.1.2 MAUs on different logical networks

The LNA can in principle listen for MAU connections on different logical networks. Depending upon protocol conformance class (see IEC 61162-400), the LNA may listen for local MAUs on the following network types:

- local program loop-back (for LNA-MAU);
- local IPC (for MAUs located on the same host computer);
- non-redundant TCP/IP link (for MAUs located on LNA TP network or another network).

The LNA is required to handle all MAUs as if they were in the same namespace independently of which network it connects to the LNA.

NOTE This list can be extended arbitrarily. The LNA is not required to accept incoming requests on any network, but it should normally be designed to do so. This can be done, for example by using a data value pair (network id, network node number) as local identification for a MAU. This means that the LNA should use the concept of TP network and NNN also for MAU connections.

6.4.2 Local MAU states

The states of a local MAU (as seen from the LNA's multiplexer/demultiplexer) are defined in Figure 12.

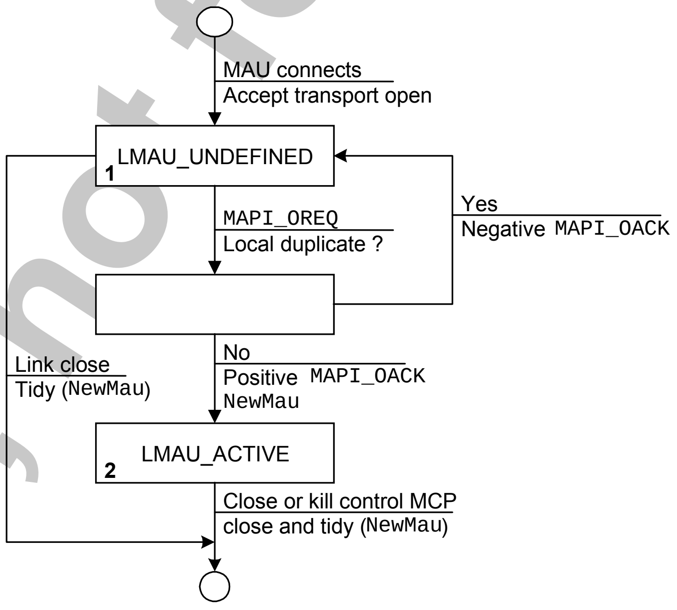

Figure 12 – MAU states seen from LNA

Local MAU management can be summarized in the following steps:

a) establish a listening port and wait for connecting MAUs. MAUs will actively try to connect to an LNA when they start and an activated LNA is required to listen. The required listening port addresses are dependent on the implementation of the MAU-LNA communication link. For TCP/IP they are defined in IEC 61162-410;

1 when a MAU connects, the LNA shall establish a data link connection to the MAU using relevant IPC or T-profile services (dependent on listening port, see previous item);

1 start MAU management using the protocol messages defined below. Deny sessions to MAUs that have a name that is already defined locally. If the session is accepted, the local MAU name database is updated (6.2 – `NewMau`). The LNA should not close the connection to the MAU immediately after denying it a session. This may cause the negative acknowledgement to be lost. To tidy up connection descriptors, the LNA can use an internal timeout or rely on the MAU closing the connection from its side;

1 service MAUs in the active state. This consists of watchdog and status control functions, various session management related services and general data transport services;

1 when the MAU management ends, tidy up network states related to the disconnected MAU, for example terminate connections to the MAU’s data objects and any pending transactions. The LNA must notify all interested LNAs and update MAU database about the death of the MAU. This is done through a call to `NewMau` in the MAU database function.

6.4.3 Starting MAU management

After a data link has been established, the following protocol messages are used to start a MAU session:

– `MAPI_OREQ`. MAU requests a session;

– `MAPI_OACK`. LNA acknowledges or denies the request.

The open request message will contain the MAU name and session control information, for example an optional watchdog interval.

The LNA is required to update the local MAU database with the new MAU immediately.

As a result of a successful session establishment, the LNA creates a special data object that can be used to control the state of the MAU. This data object is described in 6.5. If a watchdog interval is specified, the LNA is required to establish a watchdog service for the MAU using the control data object.

6.4.4 Ending MAU management

The end of MAU management is signalled by one of the following events:

– the MAU closes its end of the communication link between MAU and LNA. This may be due to a wilful action or to a communication error. The LNA is required to detect communication link close;

– the MAU itself or a remote MAU executes a kill MAU command through the control MCP. This function can be disabled for remote MAUs by supplying the MAU with a password (6.5.2);

– the MAU itself removes the control MCP (6.5);

– the MAU fails to give a positive response to a watchdog query (6.5.3).

The end of a session is handled by the LNA by closing the connection to the MAU and tidying up any state related to the communication link. The LNA shall output a message to the LNA-MAU specifying the cause when a MAU session is terminated. Note that the dead MAU will not get any message from the LNA specifying the reason for a close.

The LNA shall do the database update required by the death of the MAU (6.2). This involves also broadcasting information about the death of the MAU to all other LNAs and sending directed messages to all connected LNAs.

The local LNA shall remove all states related to MCPs or transactions associated with the dead MAU. The remote LNAs are responsible for doing the similar tidying up on their side when they receive the session close message.

6.5 MAU control MCP

6.5.1 General overview

The LNA and MAU shall automatically create a (pseudo) MCP when the MAU open message has been received and accepted by the LNA. This MCP is used to control and interrogate the state of the MAU from the LNA by using normal function MCP messages (MAPI_FREQ and MAPI_FACK). The MAU is required to answer requests to this MCP as described in this clause. The MCP identity code for this object is a constant defined as `OPEN_MCP_ID`. This MCP identity code is reserved and cannot be used by the MAU for other MCPs. The MAU control MCP is a special MCP, i.e. it is not associated with an interface.

The accept MCP is used by the LNA as if the LNA was a connected MAU. The MCP cannot be connected to by another MAU (except through the LNA MAU). Messages from the LNA to the MAU control MCP shall use the session code zero (6.6).

6.5.2 Functionality

The data object is of the function type. The MAU will get a 32 bit unsigned integer function code as input and shall return a 32 bit signed integer status code.

The set of defined input codes for the function is:

a) `M_CLOSE:` MAU is requested to close. The MAU should return non-zero if it does not close;
b) `M_RESET:` MAU is requested to close and restart. The MAU should return non-zero if it does not reset;
c) `M_KILL:` the LNA is ordered to close the connection to the MAU regardless of the return value from the MAU. The LNA may do this immediately by closing the communication link. The return value is ignored;
d) `M_STATUS:` the MAU is requested to return its status. This request is used by the LNA when the MAU watch-dog function is enabled. The MAU must return zero if it is functioning normally. A non-zero return value shall cause the LNA to close the MAU.

All these codes shall be converted by the MAU API to the corresponding event codes listed in 5.2.4. Note that the `EV_LNA_RESTART` is an internal code used by the MAU API to signal that communication links are closed and that it is safe to retry connecting to the LNA. The return code from the MAU session call-back routine shall normally be passed back to the LNA.

The LNA shall close the MAU communication link if the MAU returns affirmative to the close or reset commands. This operation should normally not be necessary as the MAU should do the closure itself.

6.5.3 Watchdog service

The LNA can be instructed by the MAU to maintain a watchdog service for the MAU. The watchdog field in the `MAPI_OREQ` message is used for this.

If enabled, this service shall function as follows:

a) at each watchdog interval, the LNA sends a status request message to the MAU control MCP;

1 if no positive reply (i.e. no message or a non-zero return value) has been obtained since the last request was sent, the MAU shall be forced closed.

Status request messages from LNA to MAU shall use a normal control data object function request. This means that the MAU cannot distinguish LNA initiated requests from requests from other MAUs, except by checking the session code message.

6.6 Session management

6.6.1 General principles

A session is an association between two MAUs that have established MCP connections between them. The LNA shall maintain a list of sessions which can be identified by the MAU. All sessions shall be uniquely identified both with respect to the identity of the participating MAUs and with respect to temporal relationship, i.e. the termination of one session and a later re-establishment of a connection between the same two MAUs shall result in a new session code being used. Session code zero is reserved for the MAU-LNA session.

It is recommended that the session code is an integer running from one and upwards and incremented at each new session establishment. As the session code is a 24 bit long integer, this should give a sufficient degree of uniqueness. The MAU needs to reset its internal session codes each time it establishes a new connection to its LNA.

All messages between two MAUs shall use the same session code. The session code is maintained by the local LNA and need not be the same for local and remote LNAs.

6.6.2 MAU session information

The session code is part of any message from the LNA to the MAU. It shall normally be extracted from the message by the MAPI and made available to the application through some form of MAPI call, for example as an interface or MCP attribute (see Table 7 and Table 8).

In the session code, the LNA is responsible for informing the MAU about the following:

a) which session a given MAU-LNA message belongs to. This is given by the numeric value of the session code;

b) the closing down of a session. This signals that a remote MAU has closed (`MAPI_SESSION`);

c) congestion state on a communication link associated with a session. The states are no congestion, congestion for low or congestion for normal priority messages (`MAPI_SESSION`).

The session code (item a)) is embedded in the basic MAU-LNA message format (8.2). The session close and the congestion information (items b) and c)) are transmitted with the `MAPI_SESSION` message. The MAU API must convert the message to one of the appropriate event codes (5.2.4).

6.6.3 Session codes used for authentication

The value of the session code can be used to verify that two messages on two different or on the same MCP have the same source MAU. This is done by checking that the values of the two session code fields are the same. By using various combinations of MCPs, this mechanism can be used both for operator and console authentication. See the authentication related companion standards in IEC 61162-420 for more information.

The quality of the authentication mechanism will depend on the T-profile in use and on the correct use of the companion standards.

6.7 Accept type interface management

6.7.1 General overview

The relationship between the entities associated with an accept interface is shown in Figure 13.

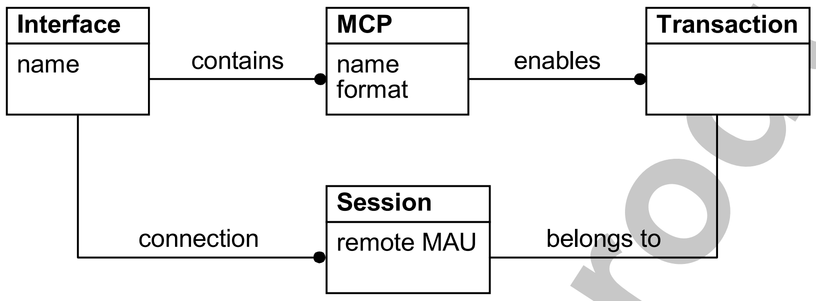

Figure 13 – Interface ER diagram

The server defined interface contains a number of MCPs. One or more client MAUs can connect to the interface (all MCPs or a subset). The successful connection defines a session (5.3) relationship between the client and server MAUs. The LNA for the client MAU shall create a session object that is associated one to one with the client interface. Likewise, the server LNA shall also create a new session object that is associated with the server interface. However, the server interface can be connected to by several clients, so there may be more than one session object to each server interface. Any transactions between the client and the server shall automatically be tagged as belonging to the respective sessions.

The definition of an accept interface can be seen as a definition of a set of interface accept MCPs. For connection management, it is necessary to look at the interface as one object, but for transaction management it is necessary to be able to distinguish between the component MCPs. These relationships should be sufficient for the LNA. The client LNA may also want to keep track of which subset of an interface a client MAU has connected to. However, the client MAU is responsible for keeping track of this.

6.7.2 Possible errors in interface definitions or connection requests

Interfaces can be defined through more than one message (from MAU or remote LNA). The general principle is that no response shall be given to an interface definition or connect request until the final message has been received (where the more flag is false). This applies regardless of whether there are errors in the definitions or not. However, this principle creates some error possibilities that are analyzed in the following subclauses.

6.7.2.1 Timeout between messages

The interface establishment times out before all messages are received. A special case is where no end message is received and there is no timeout defined. This problem can, for example be caused by a programming error that fails to send the end message.

This problem cannot be handled generally. If a timeout occurs, the timed out interface shall be negatively acknowledged (`LM_TIMEOUT`) and the LNA shall accept any other messages as a new interface definition attempt. An incomplete interface cannot be detected if no timeout is defined. This interface shall not be acknowledged, but the LNA should delete it if the involved MAU dies.

6.7.2.2 Multiple definitions on the same interface

This can happen, for example if the application or an LNA sets the more flag to false for multiple interface messages or if the application erroneously tries to define the interface in several independent messages.

This shall be handled by giving negative acknowledgements on all but the first legal definition. The negative acknowledge shall be given after all (more flag true) messages have been processed.

6.7.2.3 Inconsistent interface specifications in different messages

The interface related attribute values in different messages may differ (e.g. interface name, MAU name or password differ).

It is generally not possible to detect this error. The LNA must handle this case as if different interfaces are referenced and handle each interface individually; either as in the previous case (No final message) or as different legal interfaces. The exception is where only the password differs. This case shall be handled by negatively acknowledging the interface (`LM_PASSWORD_FAIL`).

6.7.2.4 Individual MCP errors

The MCP related attribute values in one or more messages may contain errors.

This shall be handled by waiting for the final message and returning a negative acknowledgement (`LM_INVALID_MCP`) where the bad MCPs are flagged with the respective error codes. Only the bad MCP codes are returned and only up to the number that fit in the acknowledgement message.

6.7.2.5 Cancel on incomplete interface

A cancel message may be sent before the last interface definition message has been received.

This shall cause the interface to be removed without acknowledgement to definer. Any following definition messages shall be treated as a new definition.

6.7.3 Special handling of anonymous broadcast interfaces

The interfaces identified with the name “ABCn”, where ‘n’ is a number from ‘1’ to ‘5’, are interfaces used exclusively for anonymous broadcast MCPs. The following special arrangements apply to the processing of these interface connection requests:

a) several MCPs can in principle be defined in each interface – as is normal for interface definitions. Only ABC MCPs can be defined;
b) no connection attempt shall be made. The MAU name shall be ignored for the interface definition message;
c) accept type interfaces shall cause a listening port to be opened on the appropriate ABC port. Connect type interfaces shall allow the transmission of messages on the same port;

NOTE A MAU that both listens and sends on the same MCP with the same port shall be able to read its own sent messages.

d) no state shall be saved in conjunction with the reception or transmission of messages.

The detailed processing of ABC interfaces and messages are discussed in 8.4.5.

6.7.4 Definition and removal of accept interface

The definition of an accept interface consists of defining a set of interface MCPs. The following messages are used to define or remove an interface

– `MAPI_IREQ`: MAU requests the definition of a (part of a) new interface;

– `MAPI_IACK`: LNA accepts all MCPs in request;

– `MAPI_IREM`: MAU removes the whole interface.

The LNA gets the definition message from the MAU, containing all or some of the interface components. The MAU may send more messages to define one interface, particularly in cases where message sizes are small and interfaces large. If more than one message is used, this shall be indicated with setting the *more* flag of the definition message to TRUE in all but the last message. The interface definition shall not be acknowledged before all messages have been received and processed.

NOTE The reason for the completeness requirement is to make the connection and reconnection procedure easier. By requiring that an accept interface is complete before it is established, it is easier to handle incoming client requests.

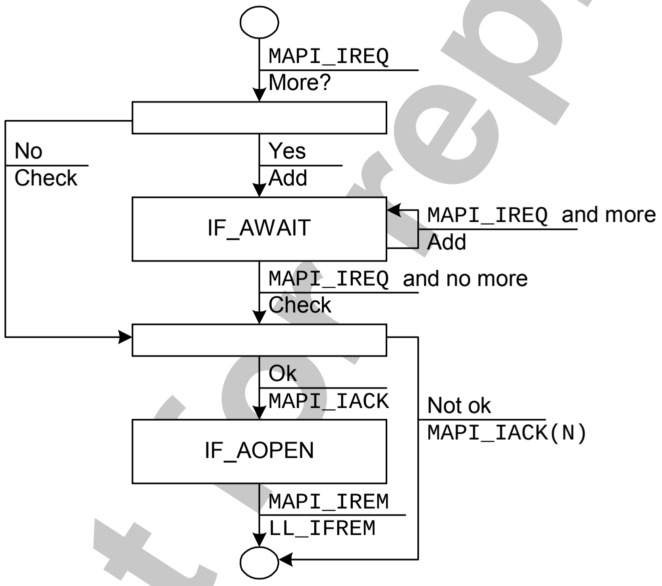

Figure 14 – State transitions for accept interface

The state transition diagram for the accept interface is shown in Figure 14.

The LNA shall accept the complete interface or deny it. In case of a denial, all detected errors shall be reported, also for individual MCPs.

If an interface is deleted by the server, a corresponding message (`LL_IFREM`) shall be sent to all connected clients. One message shall be sent for each connected MAU. This message will implicitly cancel all pending transactions.

6.7.5 Establishing and closing connections to accept interfaces

A client can establish a connection to a complete accept interface or to a part thereof. This subclause describes the handling of these connections on the server LNA side. It is assumed that the MAU identity has been established using the principles described in 6.2.

The relevant messages are:

– `LL_IFREQ`: used by remote LNA to request MCP identifiers in interface;
– `LL_IFACK`: used by local LNA to report MCP identifiers to remote LNA;
– `MAPI_IOPEN`: sent to server MAU to request client authentication and to report connection attempt;
– `MAPI_IOACK`: sent by server MAU to give or deny authentication/connection;
– `MAPI_IDOWN`: report remote close from client.

The state transition diagram is shown in Figure 15.

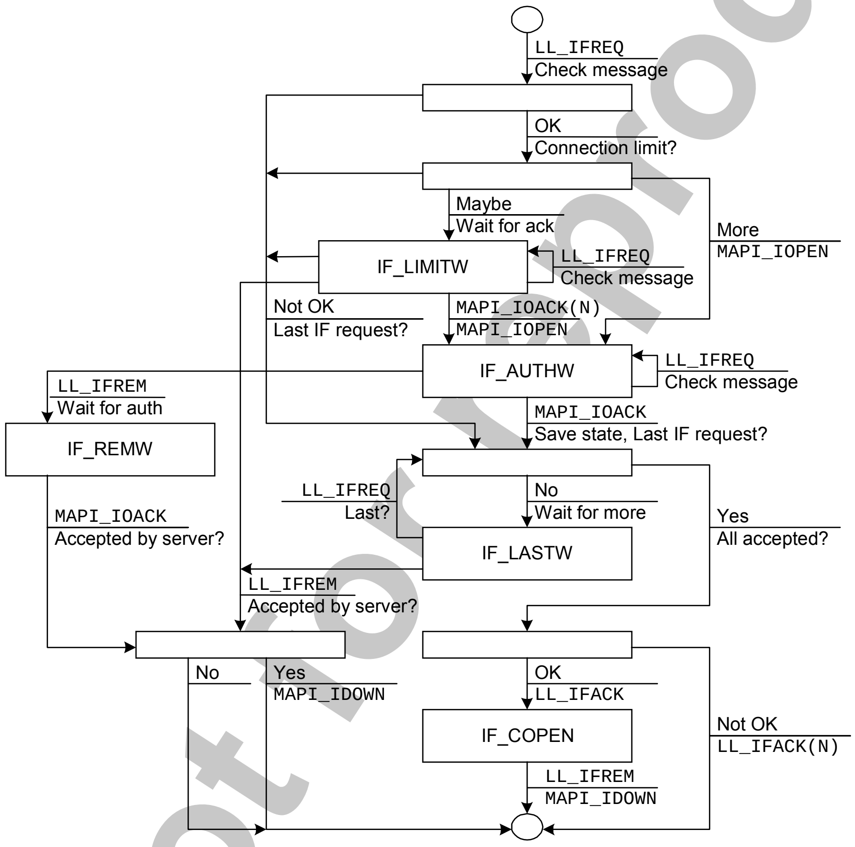

Figure 15 – Accept side interface connection states

NOTE The processing of an accept interface connect can be simplified if tests on message validity and number of connections are performed after all fragments of the connection message have been received. This will, however, create more traffic on the network for unsuccessful connection attempts. Both methods of implementation may be used and the client LNA must be able to handle both.

The remote LNA transfers an initial request with interface identity and relevant MCP attribute values to the local LNA in an `LL_IFREQ` message. The local LNA checks the correctness of the message and either immediately dismisses it if errors are detected or proceeds to check any session limits. Note that acknowledgement shall not be sent before all interface definition messages have been received (see 6.7.2).

If session limits are in force (the server has defined an upper limit of clients), this will be checked against. There are four cases to consider: 1) if a limit is in force, a client MAU with no previously established session shall be denied connection regardless of actual number of sessions already established, 2) if a limit is in force and the limit has been reached, the request shall immediately be dismissed; 3) if a limit is in force and there is already a pending request that may reach the limit on connections, the new request shall be put in a waiting state (IF_LIMITW) until the previous request was acknowledged. The result is either dismissal or further processing; 4) This request will not exceed the limit and processing can proceed.

NOTE The first case will reduce load on a server MAU in cases where session limits are used and bad-behaving MAUs try to connect to it. It will also help to remove the possibility that a bad-behaving MAU blocks another authentic client from connecting.

The next step in the processing is to put the request in a waiting state (IF_AUTHW) while the LNA interrogates the server MAU about the remote MAUs authorization. The authentication of the remote MAU (and implicitly the interface connection message) is only sent once per interface.

When the MAU session check returns, the requesting MAU is either accepted or denied a connection. In the former case, the connection is maintained until the remote MAU dies or it cancels all interface connections. Both cases are reported to the local MAU, either as a full MAU death (6.6) or as an interface connect close.

The LNA must accept a connect cancellation in any of the states. This may be sent by the remote MAU or by the remote LNA as a result of a timeout. Note, in particular, the handling of cancellation in the authentication wait state.

Multiple interface connect requests may arrive from the remote MAU. The LNA must accept these in any of the waiting state. These shall be checked for syntactic correctness and then put on hold until all messages are received and all checks are complete.

All connection request messages belonging to one interface shall normally be treated as one request with one final acknowledgement. Only one acknowledgement message is sent. In the message, only erroneously MCPs are flagged if there are individual MCP errors. If too many MCPs have errors to be fit in one message, only the first MCPs that fit are returned. This limit will depend on the T-profile message size. All aspects of the interface request messages (interface itself and all MCPs) must be correct for the interface to be accepted as a whole. Note however that a client does not connect to the complete interface. The server MAU need not remember which MCPs in an interface a client has connected to.

6.8 Connect type interface management

A client MAU can establish connections to all or to a subset of the MCPs in an interface. The connection attempt can be performed by sending one or more messages. All messages are processed as one request with one acknowledgement (possibly in more than one acknowledge message). The interface will be associated with a session as described in Figure 13.

The relevant messages are:

– `MAPI_IREQ`: used by local MAU to request an interface connection;
– `MAPI_IACK`: acknowledge a connection attempt (negative or positive);
– `MAPI_IREM`: local MAU cancels connection;
– `LL_IFREQ`: used by remote LNA to request identifiers of data objects in interface;
– `LL_IFACK`: used by local LNA to define data object identifiers;
– `LL_IFREM`: LNA cancels connection.

The connection process is described in the state transition diagram in Figure 16.

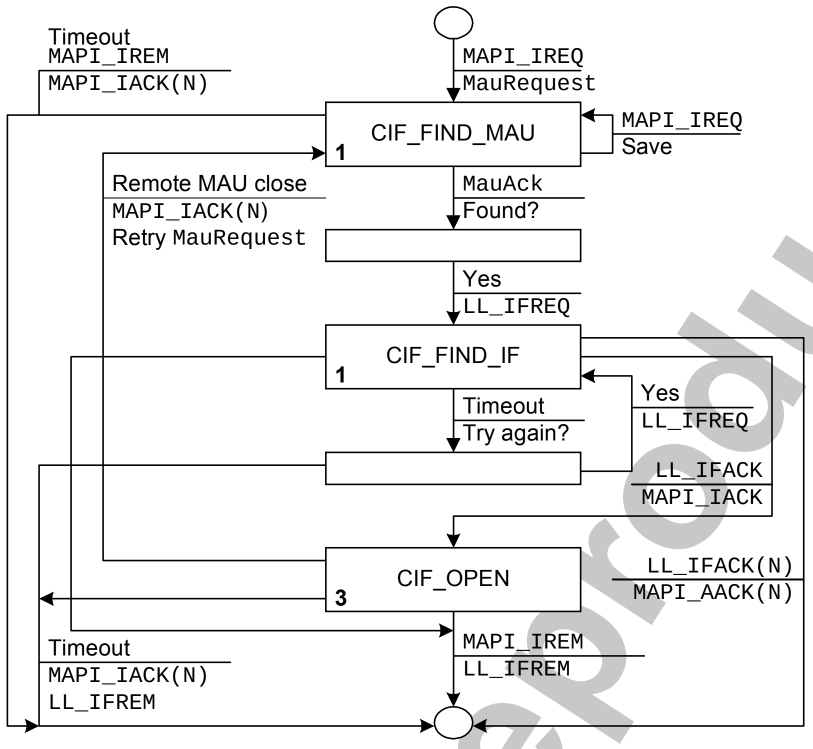

Figure 16 – Connect interface state transitions

The first step is to determine the location of the remote MAU. This is described in 6.2. When this process is complete, the connection request message is sent to the remote MAU's LNA. If the interface is reported as unknown, the request is repeated periodically until timeout. The retry periods shall be increased as follows: The first attempt shall be immediately after the remote MAU has been located. The next two attempts shall be with one second (POLL_INTERVAL) space between. The next two attempts shall be with two period intervals and thereafter the frequency shall be dropped to approximately five periods between each attempt. The client LNA is responsible for checking the timeout. It shall send an interface cancel (LL_IFREM) message to the server LNA in the case of a timeout as well as a negative acknowledgement to the local MAU.

If the local MAU sends more than one interface request, all requests shall be queued in the same waiting state. The interval timer shall be based on the first request's timeout value. Request timeout specifications following can be ignored. In any case, the client LNA must be able to receive and buffer incoming multiple definition messages from the MAU in any state (this is only drawn in the find MAU state).

NOTE It is suggested that the client LNA build a local interface descriptor structure as the messages from the MAU arrive. The client LNA may buffer all MAU requests until the last one has been received (more flag being false) before sending any messages to the server LNA. However, this increases connection duration as the communication pipeline cannot be utilized (send message, authenticate at server MAU and answer message).

The server MAU will only be allowed to export complete interfaces. The only error messages from the server LNA related to not finding an interface, is either that the complete interface is missing, in which case the retry process shall be activated, or that individual MCPs have errors or are missing. In the latter case, the client MAU shall immediately be notified of the error and no further retries shall be attempted.

The server LNA accepts or rejects the connection attempt based on all requested MCPs having been found. If the attempt was successful it will return a mapping from the local (client) MCP identity to the server's identity codes. If the attempt failed, it will return the error code for the interface as well as individual error codes for those MCPs that did contain errors. The client MAU does not need the mapping of the identity codes, but it needs the MCP error codes.

The client LNA shall inhibit multiple connections to the same interface or to MCPs within an interface by returning error. The protocol will, in general, not allow the server to distinguish between such multiple connections. This can lead to inconsistencies between client and server LNA, particularly if some connections are removed.

6.9 General transaction management

6.9.1 MCP identity and transaction address

All transactions are associated with two MCPs which again are associated with two MAUs (client and server respectively). All transactions must be addressed to the proper destination (as request to server, as acknowledgement back to client), by using the following address information:

a) destination LNA (implicit in the network node number used in T-profile);

1 destination MAU (MAU identity used on destination LNA);

1 destination MCP (MCP identity used on destination MAU).

This address information, together with the corresponding originator address information, is part of the LL_DATA message. This message has a sender and a receiver field. These reflect the sender and receiver of this particular data message, not which MAU is client and which is server.

6.9.2 Transaction identity

All transactions are assigned an identity that is maintained all the way from the client MAU, to the server MAU and back again. The server and the client MAU need not necessarily use the same transaction identity if the LNAs in between take care of correct translation. The requirement is that a message (cancel or acknowledgement) relating to some previous request shall use the same identity code as was used in the original request at each of the processing stages. Furthermore, the identity code shall, as far as possible, be different between different transactions.

NOTE Typically the client and server MAUs will use different transaction identities. The LNAs can normally use the client MAU's identity code. The server MAU must translate back and forth between own and client's code if these are different.

The anonymous broadcast uses transaction code zero.

6.9.3 Session identity

The session identity code shall follow all messages between the LNA and the MAU. The session code shall be the same for all messages originating from the same remote MAU. It shall change each time the same remote MAU disconnects and connects. It shall not be the same for two different remote MAUs.

The session code zero shall only be used for LNA originated messages (watchdog messages) and for anonymous broadcast messages.

6.9.4 Transaction types

The following table lists all transaction types supported, with their respective request and acknowledgement MAU-LNA messages.

Table 13 – Data request/acknowledge message pairs

Request type	Ref.	R/W	Request	Acknowledge	Note
Function	6.10.1	R/W	`MAPI_FREQ`	`MAPI_FACK`	
Read	6.10.1	R	`MAPI_RREQ`	`MAPI_RACK`	
Write	6.10.1	W	`MAPI_WREQ`	`MAPI_WACK`	
Non-acknowledged write	6.10.2	W	`MAPI_NREQ`	n/a	
Initial ordinary subscribe	6.10.3	R	`MAPI_SREQ`	`MAPI_FSACK`	1
Initial broadcast subscribe	6.10.3	R	`MAPI_BREQ`	`MAPI_FBACK`	1
Initial individual subscribe	6.10.4	R/W	`MAPI_DREQ`	`MAPI_FDACK`	1
Ordinary subscribe data transmission	6.10.5	R	n/a	`MAPI_SACK`	2
Broadcast subscribe data transmission	6.10.5	R	n/a	`MAPI_BACK`	2
Individual subscribe	6.10.6	R	n/a	`MAPI_DACK`	3
Transaction cancel	6.10.8	n/a	`MAPI_CREQ`	`MAPI_CACK`	4
Anonymous broadcast subscribe	6.10.7	R	`MAPI_AREQ`	`MAPI_AACK`	5

NOTE 1 This is an initial transaction giving a direct response to the request. See notes 2 and 3.

NOTE 2 These are server initiated additional responses after the initial request. The same message goes to all listeners.

NOTE 3 This is a server initiated response to individual clients. Each server message goes to one client.

NOTE 4 The cancel messages are not data messages as such, but are included here for completeness.

NOTE 5 The request goes only to the client LNA to initiate listening. The acknowledgements are sent independently of any listeners.

6.9.5 Exception handling

6.9.5.1 General error handling

The LNA is required to check incoming messages for consistency (correct receiver identities and reasonable sender identities). For incoming LNA messages the following rules apply:

a) any (typically `LL_DATA or LL_DATAC`) message to a MAU that cannot be found shall cause the LNA to send a `LL_NOMCP` message to the LNA. The data message is discarded;

b) any message to an interface or MCP that is not defined shall cause the LNA to send a `LL_NOMCP` message to the remote LNA. The data message is discarded;

c) an `LL_DATA` message with wrong transaction identity code shall be silently discarded. A warning may be printed on a diagnostic console;

d) badly formatted data messages shall be returned to the remote LNA in a local LNA generated `MAPI_xACK` message encapsulated in an `LL_DATA` message (no interface, no MCP, etc.) with the proper error code set. The incoming message shall be discarded;

e) any other erroneous messages shall be silently discarded.

For incoming MAU messages, the following rule applies:

f) errors in the MAU's request or the object's connection state shall cause a negative acknowledgement to be sent to the MAU with the proper error code set.

All errors should be reported on a diagnostic console and statistics shall be kept on the number of bad messages received. If possible, the statistics should also list the offending MAU or LNA.

Excessive errors may be handled by shutting the offending unit down by closing the communication link.

6.9.5.2 MAU or interface down

Transaction exception handling is mainly associated with the death of a remote (client or server) MAU or the closing down of a remote (accept or connect) interface.

Broken connection management consists of removing all pending transactions when the remote receiver or originator disappears. This applies to the server as well as the client side. It is not necessary to notify the local MAU of each transaction cancellation. It is sufficient to signal the close of the interface (`MAPI_IDOWN`) or the remote MAU (`MAPI_SESSION`).

6.9.5.3 Congestion and load control for transactions

The server MAU can specify a maximum number of pending transactions for an interface. This mechanism can be used to control MAU load. When enforcing this limit, the LNA shall not count transactions in the PENDING_SUBS state. Only transactions in the PENDING_ACK state shall be counted (see Figure 17).

Congestion control is done through the session control messages. This is an end to end mechanism. The LNA should take note of these messages, but, as far as possible, leave the enforcement to the sender MAUs. This means that the LNA should have enough buffer space to handle incoming messages, even on a congested link.

6.10 Accept side transaction management

The state transition diagram for transactions on an accept type MCP is shown in Figure 17. The diagram's events are labelled with the relevant message types, with the MAPI messages going to and from the local MAU and the LL messages going to and from the remote LNA.

The states are:

a) wait in interface queue: this state is used if the server MAU has defined a maximum number of unacknowledged transactions (only interface MCPs and only requests that are not urgent). If there is a maximum limit being enforced, the message is waiting for a previous message being acknowledged by the MAU (`MAPI_xACK`);

   NOTE 1 This state is not being used for congestion control. Although the sender (remote LNA or local LNA toward local MAU) should stop messages on a congested link, the local MAU must be able to receive and buffer a number of transactions that may have been sent before a congestion message was received by the remote MAU.

   NOTE 2 The number of pending transactions used to do congestion control shall not include broadcast requests that have already received its first acknowledgement.

b) pending acknowledge: this state is used for requests that have been sent to the server MAU, but have not yet been acknowledged;

c) pending subscribe: this state is used for requests that trigger multiple server initiated acknowledges (subscriptions, except anonymous broadcast subscribe). The transactions in this state are used to route later arriving server initiated acknowledgements.

Server MAU generated acknowledgement messages (on an ordinary or broadcast subscribe MCP) can legally be generated although there is no transaction object. For this particular transaction, the local MAU does not know how many, if any, subscribers there are. These messages shall be silently discarded.

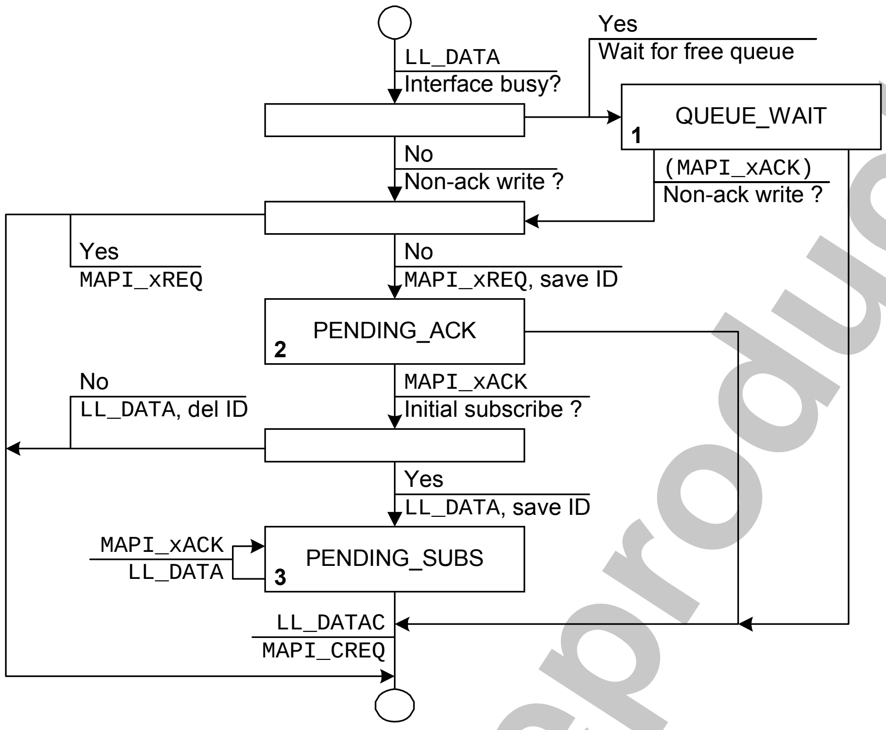

Figure 17 – State transitions for accept type data object transactions

One transaction object must be created for each incoming transaction, i.e. several transactions can be active simultaneously on the same MCP.

Transactions are only legal for MCPs in interfaces in the defined state. The interface’s transition from defined to undefined shall terminate all pending transactions (see 6.8).

Unexpected messages in any states shall be discarded with a negative acknowledge on requests where possible. In particular, the LNA must expect to receive acknowledges from the server MAU after a subscribe has been cancelled by the client.

Client MAUs, through their LNAs, can request transactions on an accept type data object they are connected to. The relevant messages are:

– LL_DATA: incoming and outgoing data messages between LNAs;
– LLBC_SACK: incoming and outgoing broadcast data messages;
– LL_DATAC: issued by client LNA to signal request or subscription termination;
– MAPI_xREQ: used from LNA to MAU for transaction requests;
– MAPI_xACK: used from MAU to LNA for transaction acknowledgements.

The requests can take several forms. Table 13 lists the available request (and corresponding acknowledgement) types.

In general, accept transaction processing for the LNA represents passing a request from a remote LNA to the local MAU and receiving the acknowledgement from the local MAU and sending it back to the remote LNA. Incoming requests and outgoing acknowledges are encapsulated in the LL_DATA message before they are transmitted between LNAs.

The LL_DATA message is an encapsulation of the MAU-LNA messages listed in Table 17. The receiving LNA needs to change MCP identity codes before it passes it to its LNA. Note that the broadcast data message is sent in an LLBC_SACK or LLBC_AACK message instead of in the normal data message.

The different types of requests must be handled somewhat differently and the LNA needs to check the type of the request to determine what kind of processing is required. The following subclauses describe the individual processing in detail. Table 13 gives a reference to the subclause in which each type of transaction is described. Note that it is necessary to check the type of transaction as well as the type of the MCP. Subscribe and write transaction message codes do not distinguish between different sub-types of data objects.

6.10.1 Read, write and function transaction handling

The following actions need to be taken by the server LNA when a normal (read, write or function) request is received from a client LNA:

a) check message for consistency (consistent receiver identities and reasonable sender identities). Discard message and return an error message to LNA if transaction request is unacceptable;

b) save client's identity (MAU, MCP and transaction identity codes) from incoming message. This information needs to be copied to the outgoing acknowledgement message;

c) strip `LL_DATA` header from message and modify MCP identity code in remaining `MAPI_xREQ` message. Add session information before it is sent to the destination MAU. The MCP identity code shall be that of server MAU's MCP;

d) continue other operations until LNA receives the corresponding `MAPI_xACK` message from the server MAU. Check this message for consistency, for example check identity and transaction codes and check that server initiated subscriptions are associated with an open MCP. Replace identity codes (MAU, MCP and transaction) in this message with those of the client MAU. Add the `LL_DATA` header with the correct identity information fields. Send message to remote LNA.

NOTE Removal of client MAU or MCP or the cancellation of the request may have caused the client's information to have been deleted before step d) is activated. In this case, the LNA shall silently discard the acknowledgement. A warning may be printed on the diagnostic console.

6.10.2 Non-acknowledged write request

The non-acknowledged write shall be handled as a normal write request with the following exceptions:

a) no change;

b) this step is not necessary since no acknowledgement is sent;

c) no change;

d) this step is not necessary since no acknowledgement is sent.

NOTE The LNA must make sure that erroneous returning acknowledgements from the server MAU (step d)) are discarded and that an error message is printed on the diagnostic console.

6.10.3 Initial subscribe

The initial subscribe request shall be handled as a normal read operation with the following exceptions:

a) no change;

b) no change;

c) no change;

d) no change, except that the client's identity must be stored until the transaction is explicitly cancelled. The identity information in the request shall be used by the LNA to build a list of subscribing client MAUs. All following server MAU initiated acknowledgements shall be sent to all subscribers on this list. The LNA shall not enter a client's identity into the list if the MAU reports a non-null status in the acknowledgement message (subscribe error).

There is no difference in how normal initial subscribe and broadcast subscribe requests are handled.

6.10.4 Initial individual subscribe

The initial individual subscribe request shall be handled as a normal function operation with the following exceptions:

a) no change;

b) no change;

c) no change;

d) no change, except that the client's transaction identity must be stored until cancelled (see previous subclause). The identity shall be used by the LNA to direct server generated acknowledgements to their correct destination.

6.10.5 Server initiated subscribe acknowledgement

The server MAU for a subscribe type data object will send unsolicited subscribe acknowledgements at any time after the object has been established. The server's LNA shall pass these acknowledgements on to the subscribing clients (6.10.3). For ordinary subscriptions (not broadcast), the operation corresponds to step d) in the normal read operation except that:

– one copy of the acknowledgement message is sent to each subscribing MAU;

– the list of pending transactions remains intact after the messages have been sent;

– the transaction identity used in the outgoing acknowledgement is the one defined in the initial subscribe request for the respective receiving MAUs. The identity code following the acknowledgement has no meaning for the LNA.

For broadcast subscription messages, only one message is sent to all receivers. Destination identity codes and transaction identities are not used. The acknowledgements are sent on the appropriate broadcast port, embedded in a LLBC_SACK message. Ordinary subscribe messages are sent to the client MAU's LNA with an ordinary LL_DATA message.

6.10.6 Server initiated individual subscribe acknowledgement

The server MAU for an individual subscribe type data object will send unsolicited subscribe acknowledgements to any selected subscriber. The server LNA needs to keep track of all pending transaction to pass the acknowledgements on to the correct subscribing client. This operation corresponds to step d) in the normal function operation except that:

– the list of pending transactions remains intact after the messages have been sent.

6.10.7 Anonymous broadcast subscribe

There is no client transaction state associated with anonymous broadcast messages. The message shall be sent out on the appropriate broadcast port embedded in an LLBC_AACK message after the message has been checked for consistency (open MCP and correct identifiers). The message must be formatted with the correct MCP identifier attributes prior to sending (hashed name – see 8.4.5). Transaction and session codes shall be set to zero.

6.10.8 Cancellation of a transaction

A client MAU can at any time cancel a transaction. The cancellation request is processed by the client LNA and results in a LL_DATAC message being sent to the server LNA. The reception of this message shall cause the server LNA to remove the client's transaction record from the MCP. No acknowledgement is sent to the client MAU or LNA.

LL_DATAC messages can be received by the server LNA without a corresponding transaction record being found (e.g. if the transaction already was acknowledged by the MAU). These messages may be silently discarded by the server LNA.

6.11 Connect side transaction management

The client LNA needs to maintain state relating to transactions which are similar to those for the accept side transactions. The state transition diagram is shown in Figure 18. The transaction object is created when a transaction request is received from the MAU and deleted when the transaction is cancelled or completed. Messages from other LNAs relating to non-existing transaction objects shall be discarded. These messages typically occur due to race conditions in transaction cancellations or timeouts and shall be treated as normal situations.

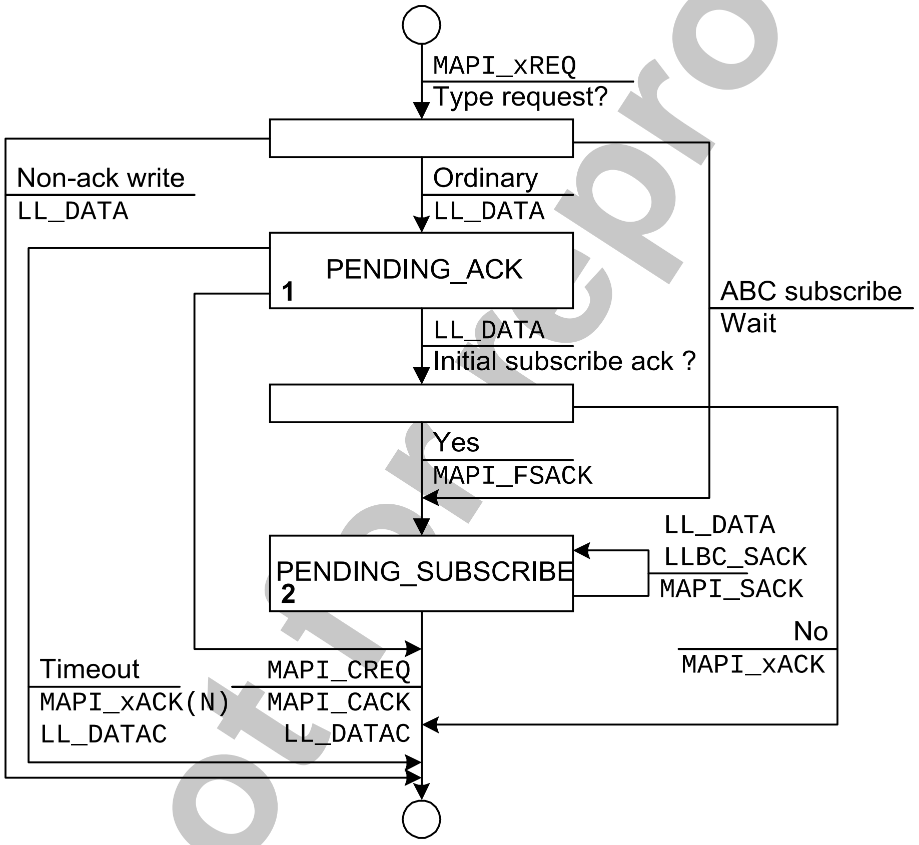

Figure 18 – State transitions for connect type transactions

The LNA's task can be described as follows:

a) the transaction request is received from the MAU as a MAPI_xREQ message. See Table 13 for the definition of the various types. Before the request can be processed, the data object connection state must be checked to see that it is ready for communication;

b) the message must be modified so that the MCP identity code is changed from the client's code to that of the server (see the MAPI_xREQ format);

c) the message must be checked for the definition of a timeout (see item f) and, if a timeout is defined, a timer must be started;

d) the request shall be embedded in an LL_DATA message (except for anonymous broadcast requests) and sent to the remote LNA;

e) different pending acknowledgement or idle states are entered depending upon the message type as shown in the figure. State (typically in the form of a transaction object) need only be saved when acknowledgements are expected;

f) a triggered timeout for the request (see item b)) shall delete the transaction object. A timeout negative acknowledge (in a MAPI_xACK message) shall be sent to the MAU and a data cancel LL_DATAC to the remote LNA;

g) a cancellation message (MAPI_CREQ) can sometimes be received in the idle transaction state. In this case, a negative acknowledgement shall immediately be returned. Otherwise, the cancel shall delete the transaction object and a cancellation acknowledge (MAPI_CACK) shall be sent to the MAU. An LL_DATAC shall be sent to the remote LNA;

h) the acknowledgement from the remote MAU (in a LL_DATA message from the LNA) shall be checked with regard to the MCP codes and the transaction serial number. A data acknowledge relating to a cancelled or timed out request (in the idle state), shall be ignored;

i) the MAPI_xACK message shall be extracted and the server MAU's MCP identity code exchanged with the local MAU's MCP identity code (see item b). The message shall be passed on to the MAU and the transaction state shall be changed as shown in the figure.

NOTE The client LNA need not modify the MAU message embedded in the LL_DATA message. Necessary identity code changes shall be made by the server LNA.

The anonymous broadcast subscribe request shall cause the MCP to enter the subscription data wait state without any message being sent.

An LL_DATA message containing MAPI_SACK (ordinary subscribe – not first subscribe) in the pending acknowledge state shall be silently discarded. This can be expected to happen for a broadcast subscribe MCP, since the two message types use different communication channels.

6.12 LNA-MAU

Diagnostics will be done by a dedicated MAU associated with each LNA. This MAU can report configuration and other management related information to some central management MAU. The LNA MAU caters to the logical network (functionality-oriented – number of MAUs, number of MCPs, MAU level traffic patterns, etc.).

It shall be possible to report the following diagnostics:

– the identity and version code for the LNA;
– the reason for closing down a local MAU or a link to a remote LNA;
– any communication errors on links to MAUs or LNAs;
– discarded and unexpected messages.

It shall also be possible to provide the following functions on local MAUs:

– send status, close, reset or kill commands to the MAU.

Performing these operations shall be dependent on the client MAU having been authorized to do the operation, for example through the authentication companion standard IEC 61162-420 or a general password or originator check mechanism.

The detailed specification of the LNA MAU will be provided by IEC 61162-420.

6.13 Use of priority levels in LNA

All LNA-LNA and LNA-MAU management messages, i.e. all messages except those that carry application data, shall use the normal priority communication link. Data-carrying messages shall use the priority level of the message itself. The message priority is specified by the client MAU at connection establishment and the server LNA shall update the messages to and from MAUs so that this priority is encoded in *priority* field of the MAU part of the message.

Note that for normal subscribes, where the LNA duplicates messages from the server MAU to all subscribing clients, the LNA may need to use different priorities for different destinations.

6.14 Congestion control

The T-profile will normally have mechanisms for signalling congestion on one or more links between LNAs or between LNA and MAU. Congestion shall be reported to afflicted MAUs by the LNAs. Congestion shall normally only afflict normal or low priority communication links. A congestion on an urgent link shall be handled as gracefully as possible, but is in principle a system design error.

The handling of congestion is designed so that the MAU normally should not see congestion as cancellation of messages (except as a result of a timeout). This also applies to acknowledgements, which will also be delivered unless the timeout triggers. The handling is also designed to minimize traffic by sending one congestion message between LNAs instead of a number of cancelled messages.

Congestion control is only used for interfaces. For MAUs that only support full MCPs, the only possible method for congestion control is to close down the MAU that causes congestion.

Congestion warnings to MAUs are warnings and do not in themselves require any particular action. However, ignored congestion warnings can cause some messages to be cancelled or the link to be closed.

6.14.1 Congestion cases

Congestion can occur in the following cases:

a) a MAU is not able to process incoming messages fast enough. The local LNA shall have a certain buffer capacity, but at some point the LNA will have to signal congestion to sending entities. This is achieved by sending a congestion message (`LL_SESSION`) to the connected LNAs. When the congestion stops, a congestion cancel is sent to the same entities with the same message;
b) a MAU is producing outgoing messages too fast for the local LNA to keep up. It will be necessary for the MAU to handle this locally, possibly by giving congestion messages back to itself. The LNA can also send a `MAPI_SESSION` message (on session zero) to the MAU which should have the same effect;
c) a remote LNA is not able to process incoming messages fast enough. The local LNA must generate the `MAPI_SESSION` message to local MAUs that communicate via this LNA;
d) a MAU determines that it cannot process any more messages from one particular remote MAU. It can then issue a congestion control message on the relevant session.

6.14.2 LNA requirements

The LNA is required to have a low and high water mark on its buffers (internal or those that are maintained by the T-profile) so that there is reasonable hysteresis in the congestion control mechanism.

The LNA shall not in general return messages unprocessed (with `LL_SESSION` flag set) in cases where congestion occurs. It shall use the congestion control mechanisms instead. This applies even when messages arrive after the congestion message has been sent. The high water mark in buffers (spare capacity) should be able to accommodate this.

The LNA that receives a congestion control message can either pass the message to the MAU (`MAPI_SESSION`) or use it to buffer outgoing messages. The LNA should in no case cancel messages due to congestion.

Normal message timeout shall be applied even in cases of congestion. Message cancellation should normally be sent to congested LNAs so that buffer space can be freed.

6.14.3 MAU requirements

If a congestion control message has been received, the MAU shall not send more messages of the congested type (low and/or normal priority) until a congestion cancel message has arrived – this is not enforced by the LNA.

7 Protocol defined as sequence diagrams

This clause defines the A-profile protocol as a set of sequence diagrams. Clause 8 defines the message formats and other details regarding functionality.

7.1 General conventions

7.1.1 Broken connections

All modules shall be designed to handle broken connection at any time in the communication sequences. All state relating to afflicted data objects shall be removed and proper negative acknowledges shall be sent to any requesting modules still alive. The default action is to assume the far side of the broken connection dead and take appropriate actions at local side.

NOTE For communication between LNAs, there is normally more than one link. A broken connection must consider the state of all links as described in 6.3.3.1.

For the various modules this means:

a) **MAU-LNA**: when the MAU-LNA link breaks the MAU shall regard all system knowledge and state relating to itself as lost. It should restart from the point of connecting to the LNA and try to establish a new connection as rapidly as possible;
b) **LNA-MAU**: when a local MAU disappears all states relating to that MAU shall be removed from the LNA. The MAU shall be deleted from the MAU database. The remote LNAs with connections to the removed MAU shall be notified (see 6.2);
c) **LNA-LNA**: when a remote LNA disappears all states relating to that LNA and its MAUs shall be removed. Local MAUs with connections to any of the removed MAUs shall be notified (see 6.3).

7.1.2 Exception handling

Most sequence diagrams are based on complete request/acknowledge message passing sequences. Various exceptions are not normally drawn into the diagram. Sometimes the exceptions are described in other diagrams, but in general, the designer needs to consider the following exceptions:

a) a cancellation of the original request. This is normally described in other diagrams, but these diagrams must be merged with the normal request/acknowledge diagrams and compared to the state transition diagrams described in the previous main clauses;
b) closing down of remote units (LNA, MAU, interface or MCP) at any time in the sequence. This may or may not be covered by the diagrams, but the general handling should be the same as for not being able to find the module in question;
c) closing down of the requesting unit. This can be handled as a cancellation, except that the dead unit needs no acknowledgement. Note also that the general MAU and LNA management systems will report deaths of this kind in any case. The designer needs to consider the handling of cancellations and death messages in a consistent manner.

Timeouts are normally included in the diagrams.

7.2 LNA management

7.2.1 Opening LNA-LNA connection

The LNAs when opening a connection between them use a handshake protocol to exchange information about each other. This protocol is implemented by the T-profile. The T-profile will report the remote network node number and other network related information once a connection has been successfully established.

7.2.2 LNA watchdog functionality

The LNA is required to send regular watchdog messages and to listen for other LNA's watchdog messages. A missing message shall be investigated by sending a link liveness request (6.3.4).

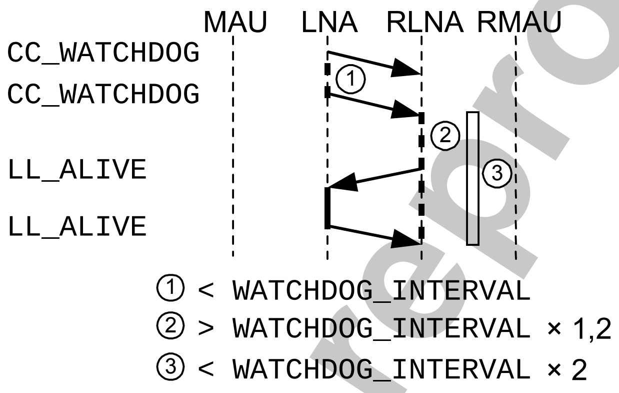

Figure 19 – LNA watchdog

The LNA shall send the watchdog message at least each WATCHDOG_INTERVAL. A missing message after 20 % over the expected time shall result in the issue of an LL_ALIVE request from other LNAs. This must be replied to by an LL_ALIVE, but the remote LNA shall also accept a CC_WATCHDOG as sign of liveness. If the remote LNA does not get an answer after twice the watchdog interval from the last sign of life, the LNA shall be defined as dead.

NOTE This procedure is only used by LNAs that maintain connections between them. The procedure replaces the link level liveness check in the previous issue of the standard.

7.2.3 Congestion control

Congestion control is done by the sending MAU when it is being notified of a too high message rate. Different congestion control messages can be sent as indicated by the comments to figure 20:

(1) from a MAU to another MAU. This results in exactly one message on each of the links shown in the diagram;

(2) from a MAU to its LNA. This shall result in the LNA sending individual session control messages to all LNAs that have MAUs connected to the afflicted MAU;

(3) from an LNA to another LNA. This shall cause the receiving LNA to send congestion control messages to all local MAUs that communicate with MAUs on the congested LNA;

(4) The receiving LNA shall duplicate all received messages to the involved MAUs.

The diagram below shows the transmission of congestion control messages.

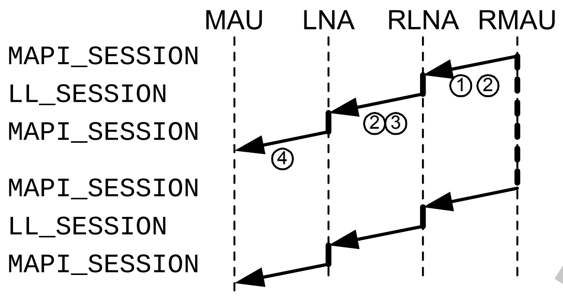

Figure 20 – Congestion messages

The MAU getting a congestion message shall stop (or start) sending messages of the specified priority to the afflicted MAU (or LNA). Continuing to send data on a congested link may cause link failure or individual message cancellations. The LNA is not expected to react to congestion control messages.

7.3 Opening and closing MAU sessions

7.3.1 MAU to MAU communication via same LNA

This standard requires that a MAU shall be handled in the same manner whether it communicates with itself, another MAU on the local LNA or a MAU on a remote LNA. For local communication only adherence to the sequence and format of messages between MAU and LNA is the requirement. In this case, the rest of the sequence diagrams can be regarded as informal information only.

NOTE Note, in particular, the LNA-MAU that is residing in the same program space as the LNA. See also 4.2.

The designer shall take special precautions with regard to broadcast communication where some T-profiles may be unable to repeat back to the sender the sent message, even if the sender was listening in on the destination port.

NOTE The LNA designer is free to decide how to handle LNA internal message flow when two MAUs both connected to the local LNA communicate. For practical reasons, it will usually be convenient to create some kind of loop-back communication channel inside the LNA. This means that the information in the sequence diagrams can be applied in full, regardless of MAU topology.

7.3.2 Opening a MAU session

The opening of a MAU is initiated from the MAU itself after the physical connection has been established. Figure 21 shows the message sequence. The MAU requests a session opening through the `MAPI_OREQ` message and gets the acknowledgement (or denial) through the `MAPI_OACK` message. The LNA registers the MAU in the MAU database and broadcasts information about it through `CC_DEFMAU`.

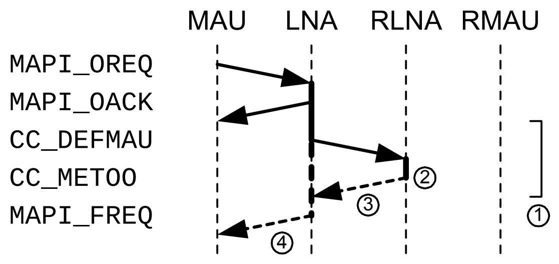

Figure 21 – MAU session open

The following comments to the sequence diagram apply:

(1) the LNA shall deny the MAU session (negative acknowledge in the `MAPI_OACK` message) if an equally named MAU already is registered at the local LNA. In this case, sequence 1 is omitted;

(2) if any remote LNA has a *non-local* duplicate of a newly advertised MAU, the old MAU reference shall be deleted and replaced by the new MAU reference;

(3) a `CC_DEFMAU` message received by a remote LNA which already has registered a local MAU with the same name shall cause the remote LNA to send a `CC_METOO` message, but retain the local MAU in its database;

(4) the LNA shall periodically interrogate the state of the MAU, if instructed to do so, through a watchdog field in the open message. The watchdog function is performed on the MAU control data object (see below).

The successful opening of a MAU session causes a control data object to be defined for the MAU. This data object is used to interrogate the MAU of its status and also to close the MAU down if desired. The data object is also used by the MAU watchdog function implemented by the LNA. The data object is referenced by the object code `OPEN_MCP_ID`. See 7.3.3 and 8.2.2 for more details. The definition of a new MAU will enable the establishment of connect interfaces (7.4.3).

7.3.3 Closing a MAU session

The closing of a MAU session can be caused by different events. In all cases, the physical connection either has been closed as one of the direct effects of the event or when the LNA immediately closes the connection when the event is detected.

The different session close cases are:

a) the MAU explicitly closes the connection at its side to signal a session end. This is the normal way to signal a MAU side session close;

b) the connection closes because of a communication error detected on the LNA side, for example a garbled message or a transport layer error. This shall, if possible, be reported on a diagnostic console or to the LNA MAU;

c) the LNA closes the connection because it received a `M_KILL` command or because the MAU did not positively acknowledge the LNA's `M_STATUS` request on the `OPEN_MCP_ID` (see 6.5 for details);

d) the MAU's session establishment timer times out (see 7.3.2).

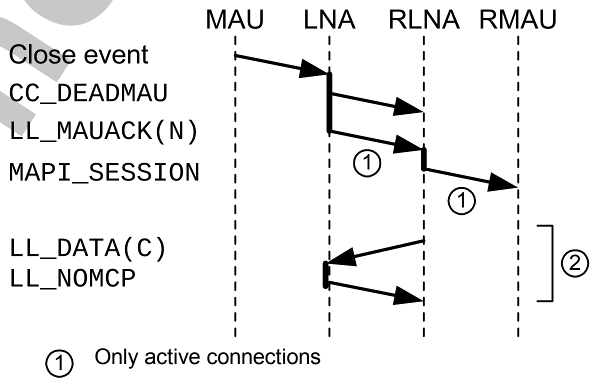

Figure 22 – Closing a MAU connection

In all cases, the LNA immediately closes the physical connection and notifies the rest of the system of the disappearance of the MAU. LNAs with connections to interfaces on the removed MAU (comment (1)) shall be directly notified through the use of `LL_MAUACK` messages (see 7.3.5 for handling of this message on the client side). The rest of the system is notified through broadcasts from the LNA (`CC_DEADMAU`).

Any remote MAU that has a connection to the removed MAU (comment (1)) shall be notified about the death by its local LNA through a session-down message. The LNA shall then automatically try to re-establish the connection. This process is described in 7.4.

If the remote LNA sends data (`LL_DATA`) or cancellation (`LL_DATAC`) messages after the removal of a MAU (the MAU is not known locally) an `LL_NOMCP` message shall be sent to the offending LNA (comment (2)). This is normally caused by the data message already being in the pipeline when the local MAU dies. However, it may also indicate a logical error in the MAU or the LNA.

7.3.4 Finding a remote MAU in LNA

The LNA shall maintain a list of local and remote MAUs that are known to it (6.2). It shall also provide a MAU connection service. To facilitate this service, the LNA uses a mix of broadcast general name look-up and directed point to point connection establishment requests.

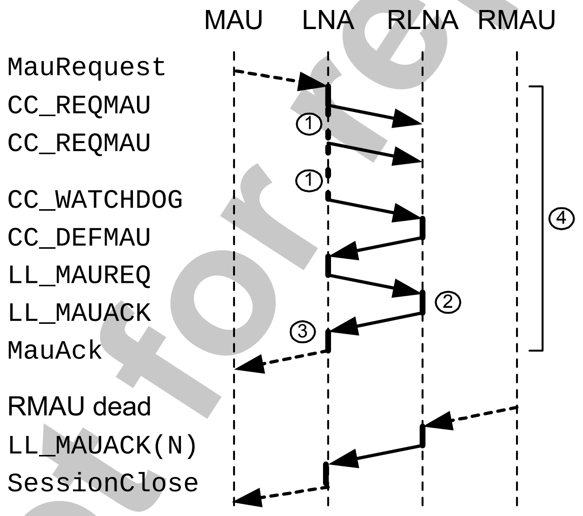

Figure 23 – MAU look up and close

The LNA's MAU look up module will get requests for MAUs as described in 6.2. It is required to determine the location of the MAU. This is done as indicated in the above figure and in the referenced clause. The following comments apply:

(1) unknown MAUs shall be looked for by increasing intervals, either through one-by-one name requests or through the general LNA watchdog mechanism;

(2) the location of a MAU reported through broadcasts shall be verified by a reliable link exchange;

(3) if the verification fails, the attempt shall be retried by sending new broadcasts;

(4) if there is already a connection to the MAU, the session establishment steps shall be dropped. There can also be a timeout associated with the request that can cause a negative acknowledgement to be delivered at this point.

The module shall also notify the caller of a close in the MAU connection (session close). It is the responsibility of the caller to request a new look-up for the MAU after a close.

NOTE The labels MauRequest, MauAck and SessionClose are services provided by the LNA itself and are not messages.

An LNA to LNA connection may have to be established before the first contact between the local and the remote LNA (see 7.1.2).

The client LNA is required to try to restore any interface connection that goes down due to a server failure. The connection attempt will start as soon as the client MAU has been notified of the exception. All timeouts and other parameters originally defined by the client MAU will be used in the new connection attempt. The client MAU can cancel the attempt by issuing a close interface command.

The following subclause, 7.3.5, defines the actions to be taken in the face of a set of expected connection down events. Following the specified link down actions, the LNA shall try to reconnect missing connections from the MAU search initiation.

7.3.5 Server MAU or LNA dies

The connection between the server LNA and the server MAU may go down. In this case, the server LNA reports the death of the MAU to all LNAs that are known to maintain connections to the dead MAU (see comment (1) below). Only one message need be sent to each LNA.

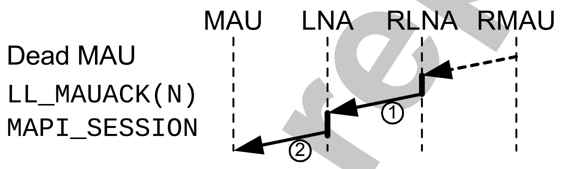

Figure 24 – Server MAU dies

The LNA reports the loss of connection to all MAUs that have interfaces associated with the dead MAU (MAPI_SESSION). Then the LNA starts the attempt to reopen all connections. The re-opening is equivalent to the normal open procedure except that the initial MAU request message is unnecessary.

See also previous subclause on the handling of the remote MAU information in the LNA.

The following comments apply:

(1) one message for each remote connected LNA, independent of MAUs served by LNA. Alternatively, this message may be substituted with a close of the LNA-RLNA link;

(2) one message for each MAU, independent of the number of interfaces.

The LNAs (and MAU) shall clean up all states related to the lost MAU, for example connections and pending transactions.

7.4 Opening and closing interfaces

7.4.1 Opening accept interface

The opening of an accept IF causes the server defined attributes for all the MCPs in the interface to be registered in the local LNA. The sequence diagram is shown in Figure 25.

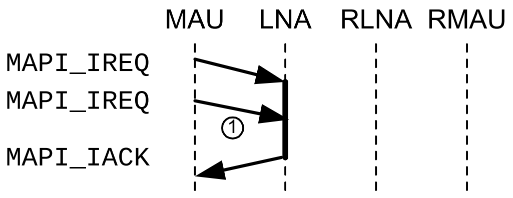

Figure 25 – Accept IF open

Any number of definition messages may be sent by the MAU before the final request message (signalled with false in the more field) is sent. The LNA shall only acknowledge after the final message has been sent (comment (1)). The LNA may deny the request (negative acknowledge) if errors are detected in the message.

7.4.2 Closing accept IF

The semantics of the interface requires that the server MAU and LNA keep track of client connections. This requires notification when an accept interface goes down. The sequence diagram is shown in Figure 26.

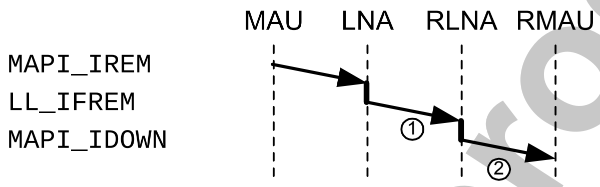

Figure 26 – Closing accept interface

The following comments apply:

(1) all client MAUs shall be notified, i.e. more than one message to each connected LNA may be necessary;

(2) the remote LNA shall inform the client MAUs about the interface close and cancel all pending transactions without further notification.

All pending transactions shall be cancelled without notice, also locally.

7.4.3 Opening connect IF

The open connect IF sequence is shown in Figure 27.

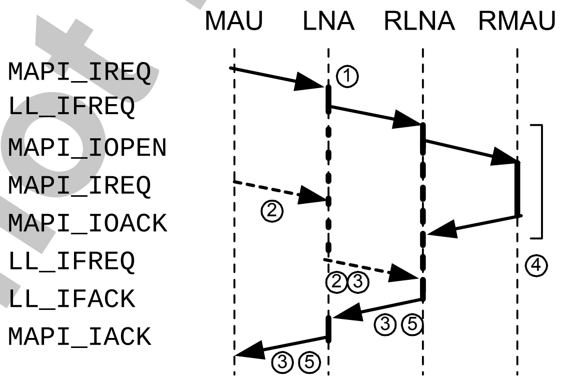

Figure 27 – Connect IF open

The following comments apply:

(1) after the first connection request, the LNA must first locate the RMAU. This is achieved by using the principles discussed in 7.3.4. This operation may or may not require communication;

(2) the MAU may issue more than one request (see the state transition diagrams in 6.7.5 and 6.8). The client LNA can buffer requests, although it is recommended that messages be immediately forwarded. Dependent on message size limitations, the LNA may also have to fragment each MAU request into several LNA-LNA requests;

(3) the LNA and the RLNA shall not give an acknowledgement before all requests have been received (the more flag is false). If the request times out on the client side, the client LNA will cancel the request;

(4) the RLNA may have to delay the authentication request to the RMAU if there is a connection limit in force and the limit has been reached by still pending requests. Normally, the authentication request to the RMAU will be sent immediately after receiving the first request from the LNA. The client authentication and connection announcement between RLNA and RMAU is done only once per interface and connection;

(5) more than one acknowledgement message may be needed if message lengths are small. The last message is signalled with the more flag being false.

The RLNA need not store state on the actual MCPs the client has connected to within an interface. This information is maintained by the LNA, based on MCP identity information from RLNA. Another state is maintained only on an interface and transaction basis.

The LNAs shall insert a session code identifying the MAU-MAU session in all messages to its MAUs. This session code is also used in data messages.

7.4.4 Closing connect interface

A client initiated close removes all connections to MCPs in the interface. All pending transactions are cancelled implicitly and without notification. Later data acknowledges can be silently discarded.

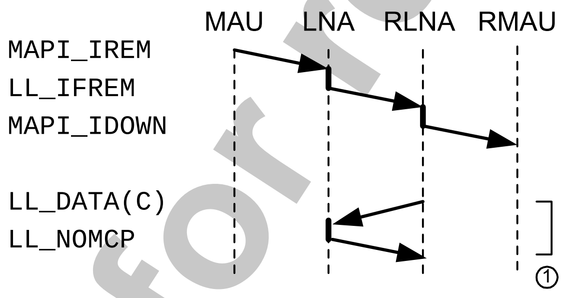

Figure 28 – Interface close

If the RLNA sends further data messages (`LL_DATA` or `LL_DATAC`), these shall be answered by `LL_NOMCP`. This typically means that a data message crossed the `MAPI_IREM` message somewhere in the communication pipeline (comment (1)).

7.5 Data transfer messages

The data transfer messages move data-requests from one MAU to another and transports the acknowledgement in the other direction. These messages are normally sent on virtual connections between one connect and a corresponding accept MCP. The exception from this is the broadcast messages.

The basic message sequence is described in 7.5.1. It applies to bi-directional data transfers where the client explicitly initiates each transaction. The client can initiate a new request on a MCP before the next is acknowledged. Each transaction must be looked at as a separate object with a unique identity.

Other data transfer types are unidirectional. This applies to subscribe and non-acknowledged write. Broadcast subscribe is a special case of the subscribe where network bandwidth is saved by using broadcasts for data transfers. The trade-off is a certain chance that messages may be lost.

7.5.1 Data transfer of read, write or function type

MCPs that support read, write or function, basically support the same transaction mechanism. This is a full bi-directional client-initiated transfer. The difference between read, write and function is that the read only carries data from the server, the write only carries data to the server and the function carries data both ways. The message sequences are the same in all three cases. The sequence diagram is shown in Figure 29.

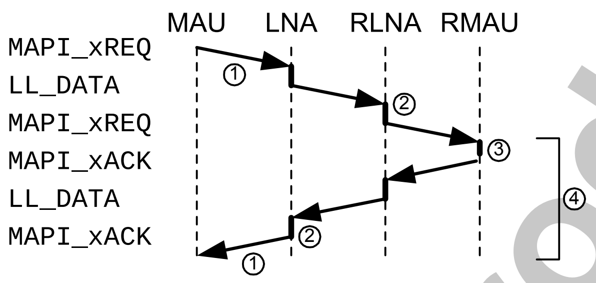

Figure 29 – Normal data transfer sequence

The sequence is initiated by the client MAU and propagated forward to the remote MAU and back again through the system. Several transactions can be active at any one time. Each transaction is handled individually.

The following comments apply:

(1) each message is equipped with a transaction identity. This identity shall be restored by all parties when the corresponding acknowledgement is sent back, for example the local LNA must save the client MAU’s transaction identity and restore it when the acknowledgement is sent back to the client. This identity code is also used for cancellations and multiple acknowledgements (subscriptions). The transaction identity is not useful for non-acknowledged transactions;

(2) the LNAs shall insert a session code in all messages to the MAU that identifies the MAU-MAU session in force. The same session code shall be used in the interface connection establishment. The message may be queued in the LNA if the MAU is congested or if there is a limit on the number of pending transactions the MAU accepts;

(3) the remote MAU need not acknowledge the requests in any particular sequence and it may delay acknowledgements indefinitely;

(4) for non-acknowledged write transactions, the LNA and RLNA need not keep track of the transactions and the RMAU shall not acknowledge the request. The session information shall be inserted in the message to the RMAU.

The various message types are listed in Table 14.

Table 14 – Transaction message types

Generic	Read	Write	Non-ack write	Function
MAPI_xREQ	MAPI_RREQ	MAPI_WREQ	MAPI_NREQ	MAPI_FREQ
MAPI_xACK	MAPI_RACK	MAPI_WACK		MAPI_FACK

Between the LNAs, the LL_DATA message shall be used for all transaction types.

A timeout may be associated with any request. This timeout shall be enforced by the client LNA and shall, if triggered, result in a negative acknowledgement message to the client MAU and a cancellation message to the RLNA and RMAU (see 7.5.3). Any errors during processing of a request shall result in a negative acknowledgement to the originator LNA and/or MAU. The negative acknowledgement shall be encapsulated in the proper message type, but the data field shall be empty.

7.5.2 Subscribe data transfers

Subscribe data transfers differ from the basic type in that several acknowledgements may result from one request. This is typically done by returning one initial acknowledgement as a direct response to the initial request and then generating later acknowledgements at the server MAU's discretion. There are four different types of subscriptions:

a) ordinary subscribe: the initial acknowledgement is sent directly and only to the requesting client. Later, the RMAU at self-controlled events, generates one acknowledgement message, referenced to its accept MCP that is duplicated by the RLNA to all subscribing clients. The duplication shall copy the client's transaction code into the message;
b) individual subscribe: this is similar to ordinary subscribe, except that the RLNA does not duplicate later acknowledgements. The RMAU generates individual later acknowledgements to each client;
c) broadcast subscribe: as ordinary subscribe, but the later acknowledgement is sent as one message on the broadcast channel `ABC0` to all listening clients. The LNA must copy the correct transaction code into the message;
d) anonymous broadcast. There is no remote MAU in this mechanism. The sender sends to a global broadcast address (`ABC0` to `ABC5`) and receivers listen to the same.

The sequence diagram is shown below.

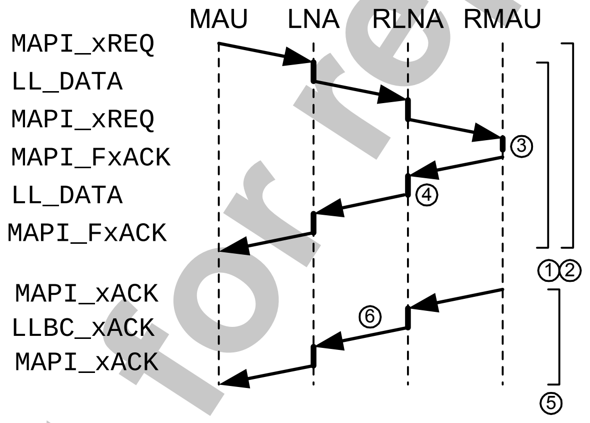

Figure 30 – Subscribe data transfer

The first sequence (group (2)) describes the initial request. The second sequence (group (5)) describes the later acknowledgements.

NOTE This standard does not prohibit the transmission of more than one initial request. This shall in principle register one pending transaction at the RLNA for each request and thus, cause multiple later acknowledgements. This has no use for ordinary and broadcast subscriptions, but may be useful for individual subscriptions.

The message codes shall depend on the type of subscription as defined in the following table.

Table 15 – Subscription messages

Type	Init. MAU req.	Init. MAU ack.	Later MAU ack.	LNA-LNA
Ordinary	MAPI_SREQ	MAPI_FSACK	MAPI_SACK	LL_DATA
Broadcast	MAPI_BREQ	MAPI_FBACK	MAPI_BACK	LLBC_SACK
Individual	MAPI_DREQ	MAPI_FDACK	MAPI_DACK	LL_DATA
Anonymous	MAPI_AREQ	n/a	MAPI_AACK	LLBC_AACK

The following comments apply:

(1) for anonymous broadcast requests, the message is only sent to the local LNA. The rest of the sequence is void. The local LNA will establish a listening port and will transfer messages received (comment (5))) to the listening client;

(2) otherwise, the first transaction is functionally equivalent to a normal client initiated read (ordinary and broadcast subscribe) or a function transaction (individual subscribe). See 7.5.1;

(3) the server MAU can delay the first acknowledgement or send acknowledgements in any order as for normal transactions;

(4) the server LNA needs to keep the transaction as pending although a first acknowledgement is sent to the client. The first acknowledgement is delivered only to the requesting client;

(5) the RMAU initiates all later transactions. It does this by sending `MAPI_xACK` messages to the RLNA. The time at which these messages are sent are determined solely by the server MAU. The LNA embeds the message into the correct LNA-LNA message (see Table 15);

(6) the LNA shall keep track of all active subscribing MAUs and send one message to each remote MAU (for ordinary subscribe), one broadcast message (for broadcast subscribe) or only one message to the MAU the transaction belongs to (individual subscribe).

The client MAU can stop delivery of subscription acknowledges by cancelling the subscription (see 7.5.3). This still lets the connection stay open. The client must close its MCP to remove the connection.

A timeout may be associated with the initial request. This timeout shall be enforced by the client LNA and shall, if triggered, result in a negative acknowledgement message to the client MAU and a cancellation message to the RLNA and RMAU (see 7.5.3). Any errors during processing of an initial request shall result in a negative acknowledgement to the originator LNA and/or MAU. The negative acknowledgement shall be encapsulated in the proper message type, but the data field shall be empty. Errors during processing of later acknowledgements can also result in a negative acknowledgement on the first request if appropriate. A negative acknowledgement shall, in any case, stop later acknowledgements.

7.5.3 Data transfer cancel and transfer timeout

The cancel mechanism can be used on any type of data transfer. It is used to stop the server's acknowledgement of a request. Figure 31 shows the sequence diagram for the general cancel operation. The `MAPI_xREQ` and `MAPI_xACK` messages can be any of the legal request or acknowledgement messages as defined in Table 14 or Table 15.

Cancel is initiated by the client MAU with the `MAPI_CREQ` message. The client's LNA sends the cancellation further on in the chain, while it blocks any acknowledgements that arrive from the server.

The following comments apply:

(1) a cancel must be referred to a previous request. The transaction identity code must follow the cancellation. Any request can be cancelled, although it will have no effect on non-acknowledged writes;

(2) a transaction timeout can be specified by the client MAU and shall be enforced by the LNA. If the timeout triggers, it shall have the same effect as if a cancel was issued by the MAU. However, instead of the cancel acknowledgement normally delivered when the cancel is complete, the timeout shall cause an immediate negative acknowledgement on the initial request to the MAU;

(3) if the cancellation is applied to an anonymous broadcast, no further action is necessary than cancelling the listening port. The cancel acknowledge shall be sent to the MAU;

(4) if the transaction acknowledgement was already received and passed on by the LNA (this is recognized as if there is no transaction object corresponding to the cancellation request), the MAU shall get a negative cancel acknowledgement. No further action is necessary. This corresponds to sequence (4) and voids sequence (5);

NOTE The API in the MAU must determine how to handle this case. It is usually most convenient to ignore the negative cancel acknowledgement.

(5) if the LNA did not already receive the transaction acknowledgement, it shall send the cancel request on to the remote LNA and return a positive cancellation acknowledge to the MAU. The transaction object shall be removed and later data acknowledgements discarded. This will result in sequence (5) and some parts of sequence (4) (dependent on the server's acknowledgement being intercepted or not);

NOTE A negative acknowledgement on a subscription request represents the termination of the initial request. In this case, the referenced object should be deleted and a cancellation shall be negatively acknowledged. The subscription transaction object shall survive the initial acknowledgement.

(6) if the remote LNA already has got the acknowledgement, it can ignore the cancel. Otherwise, it passes the cancel request on to the remote MAU. Note that subscription transaction objects shall normally survive the initial acknowledgement;

(7) any transaction acknowledgement sent from the remote MAU shall be intercepted and stopped as early as possible. Unless it was already passed to the MAU by the local LNA (comment (4)) it shall not be delivered.

Successful subscribe transactions (except anonymous broadcast) will always have a transaction identity to cancel. Note that any later acknowledgement shall be cancelled as well.

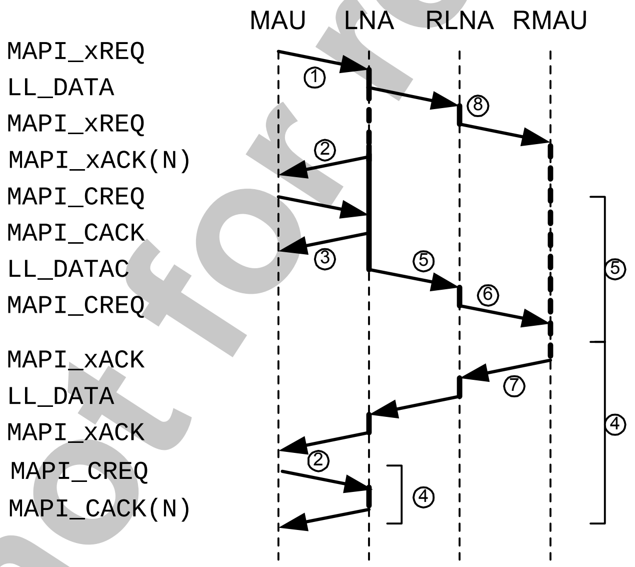

Figure 31 – Cancelling a data transfer

A cancel message cancelling side effects of the transaction cannot be relied upon, even if the cancel was delivered to the remote MAU. This is determined by the application.

8 Message definitions

8.1 Introduction

This clause describes the format of all message types used by this standard. Subclauses 8.2, 8.3 and 8.4 each describe a single communication path. The first is the MAU – LNA path. This is followed by a description of the LNA – LNA reliable message path, and finally the LNA – LNA unreliable multi-cast path.

8.1.1 Common message format

All messages, regardless of transportation path, share a common format as described in the table below. The purpose of using this format is to make it easier to reuse components in the transport layer interface (TLI) and the T-profile, and to provide possibilities for a more general handling of messages in the various software components (LNA and MAPI).

The message format is also defined so that the T-profile can use the header to do message fragmentation and assembly over, for example a stream-oriented protocol (see IEC 61162-410).

Table 16 – Common message format

Octet #	Data type	Field description
0 – 1	`word16_m`	Magic code (`MV4_MAGIC`)
2 – 3	`word16_m`	Message type
4 – 5	`word16_m`	Priority
6 – 7	`word16_m`	Message length (= 8 or x+1)
8 – x		Data

The magic code is used to identify messages from a special version of the standard or to distinguish between messages of a special class. For the purposes of the A-profile, only one code is used: `MV4_MAGIC`. Messages arriving on a link and having non-recognized magic numbers should cause the link to be closed.

The message type is an identifier for the application layer. It is dependent on meaning and context of the message.

The priority defines the priority of the message. The priority is also sometimes determined by the transport path that the message follows. The following priority codes are allowed by this standard:

a) `PT_LOW`: low priority (normally used only for stream communication);

b) `PT_NORMAL`: normal message priority;

c) `PT_URGENT`: urgent message priority.

The T-profile shall enforce a best-effort priority handling on the network. The MAU and the LNA are required to process higher priority messages before lower priority messages. Between messages on the same priority level, the units shall enforce round robin (first in first out) handling.

The message length contains information for the T-profile and the application layer. It defines the overall length of the message, including the header.

The rest of the message, if any, is application data.

8.2 MAU-LNA messages

8.2.1 Message format

All messages between MAU and LNA have the same basic format given in Table 17. The message starts with a magic number, a message type identifier and the message length. Other fields are described in the following subclauses.

Table 17 – Basic MAU – LNA message format

Octet #	Data type	Field description
0 – 1	`word16_m`	`MV4_MAGIC`
2 – 3	`word16_m`	Message type
4 – 5	`word16_m`	Priority
6 – 7	`word16_m`	Message length (= 20 or x+1)
8 – 9	`mcid_m`	MCP ID
10 – 11	`word16_m`	Session code
12 – 15	`word32_m`	Transaction identity
16 – 19	`word32_m`	*status* or *timeout*
20 – x		Data

8.2.1.1 MCP ID field.

All LNA-MAU messages shall be referenced to one of the MAU's MCPs by using the MCP's corresponding identity code in this field. This identity code is defined by the MAU during interface connection establishment. General MAU related messages use the special code `OPEN_MCP_ID` (value zero) in this field.

Note that the LNA must map incoming MCP identity codes (from a remote MAU) to the identity code used by the local MAU when remote LNA-LNA messages are processed.

8.2.1.2 Session code

The session code is used to identify the MAU-MAU session to which the message belongs. The session code is defined by the LNA. The special session code zero is used for LNA-MAU session related messages. The session code is not used in messages from MAU to LNA, except for the `MAPI_SESSION` message.

8.2.1.3 Transaction identity

The transaction identity shall be defined by the MAU for all outgoing requests. This identity code will be used by the LNA to mark the returning acknowledgement messages. Likewise, the MAU shall use the transaction identity of the incoming request when it is acknowledged.

The LNA is also required to make sure that outgoing transaction codes match those of the incoming requests. This means that the LNA must store the incoming codes, as a minimum to check consistency between incoming and outgoing messages.

8.2.1.4 Status or timeout field

The last field before the (optional) message body, *status* or *timeout*, will have different meaning depending on the type and direction of the message.

a) **Request** messages from MAU to LNA will have a *timeout* field. A timeout value of zero means forever, i.e. no timeout. A non-zero timeout value specified in milliseconds shall be read and processed by the local LNA. The LNA shall either complete the request within the specified timeout or return a negative acknowledge with the *status* code set to LM_TIMEOUT.

b) The *status* or *timeout* field shall be ignored for **request** messages going from LNA to MAU.

c) **Acknowledge** messages in either direction using the field for *status* codes (e.g. when LNA indicates a timeout – see item a). The status code is used to signal the success or failure of a request.

8.2.1.5 Status codes

The status code LM_OK (value zero) is used to signal a normal acknowledgement. Any other code specifies some kind of error. All legal codes are defined in annex B. For compatibility with future versions of the standard, all implementations shall accept unknown error codes and handle them as unspecified errors.

The MAU shall receive at most one acknowledge message for each request (except for subscriptions). In the case of an LNA internal error report or a timeout, the LNA shall filter out any other message that is an effect of the original request being processed at a remote site. This is, in particular, appropriate for transaction requests.

A message with a non-zero status field will normally have an empty data field.

8.2.2 MAU session control

8.2.2.1 Request MAU session open (*MAPI_OREQ*)

This message is sent from the MAU to the LNA to make the MAU known to the LNA. If the session can be successfully opened, the LNA shall perform the following actions:

a) allocate an LNA-unique number as identifier for the MAU (mauid_m). The LNA shall try to avoid reusing old serial numbers since this may cause confusion in other parts of the system;

b) create an implicit accept type MCP for the MAU. This MCP shall have the ID code OPEN_MCP_ID;

c) register the MAU in the system. This means registering the new MAU in the name look-up database.

The message shall be answered with a MAPI_OACK message from the LNA.

Table 18 – MAU session open request message

Octet #	Data type	Field description
0 – 1	word16_m	MV4_MAGIC
2 – 3	word16_m	MAPI_OREQ
4 – 5	word16_m	Priority
6 – 7	word16_m	Length = x +1
8 – 9	mcid_m	OPEN_MCP_ID
10 – 11	word16_m	Session (not used)
12 – 15	word32_m	Transaction identity
16 – 19	word32_m	Timeout (watchdog interval in ms)
20 – 21	word16_m	Protocol version, major number
22 – 23	word16_m	Protocol version, minor number
24 – 25	word16_m	Conformance class
26 – x	mauname_m	MAU name

The watchdog interval instructs the LNA to periodically interrogate the MAU of its status. If non-zero, the watchdog interval is specified in milliseconds. A zero in this field disables the watchdog function.

Following that are two small integers giving the version code of the protocol, major version number first (most significant) and minor version number second. The version numbers for this edition of the standard are defined in 3.1.24.

The conformance class is defined in the companion standard IEC 61162-420. It defines the class of application implemented by the MAU (see `INTERPRETATION PFClass`). The following codes are reserved by this part of the standard:

a) `PFC_NONE` (zero): no particular application class;
b) `PFC_LNA` (one): the MAU associated with the LNA (LNA MAU);
c) `PFC_SIMPLE` (two): MAU has rudimentary version control fields;
d) `PFC_FULL` (three): MAU has configuration management capabilities.

Other conformance codes may be defined by the companion standard IEC 61162-420.

The MAU name is a null terminated string. If the MAU name starts with underscore, it shall be converted to a system unique MAU name by the LNA.

8.2.2.2 Acknowledge MAU session open (*MAPI_OACK*)

The LNA shall reply to the `MAPI_OREQ` message with the message shown in Table 19.

The acknowledge message's *status* field specifies the result of the open request. The format of the returned packet will depend on the value of this field. The following codes are used:

a) `LM_OK` means that the session is open and that the MAU is registered;
b) `LM_DUPLICATE_MAU` means that the session was denied because another MAU on this LNA had the same name;
c) `LM_FORMAT_FAIL` means that some data fields in the open request were formatted incorrectly;
d) `LM_NOT_SUPPORTED` means that the protocol version specified is not supported by this LNA.

Table 19 – MAU open acknowledge message

Octet #	Data type	Field description
0 – 1	word16_m	MV4_MAGIC
2 – 3	word16_m	MAPI_OACK
4 – 5	word16_m	Priority
6 – 7	word16_m	Length = x+1
8 – 9	mcid_m	OPEN_MCP_ID
10 – 11	word16_m	Session (zero)
12 – 15	word32_m	Transaction (as in the request)
16 – 19	word32_m	*status*: error code | LM_OK
20 – 23	[2]word16_m	LNA protocol version code (major, minor)
24 – 25	[2]word16_m	LNA software version code (major minor)
26 – x	address_m	Network address of LNA

8.2.2.3 Closing the MAU session

Closing of the MAU session is done by simply closing the physical connection between the MAU and the LNA. It can also be done by sending a “session zero down” in the session control message (see next subclause).

8.2.2.4 Session control (***MAPI_SESSION***)

This message is sent or received by a MAU to indicate changes in a session. It is normally used to mark a session as congested or not congested. A MAU can also send it to request the close of its own session. The LNA will then close the MAU down.

Table 20 – Session control

Octet #	Data type	Field description
0 – 1	word16_m	MV4_MAGIC
2 – 3	word16_m	MAPI_SESSION
4 – 5	word16_m	Priority
6 – 7	word16_m	Length = 20
8 – 9	mcid_m	OPEN_MCP_ID (not used)
10 – 11	word16_m	Session
12 – 15	word32_m	Transaction identity (not used)
16 – 19	word32_m	Status contains session control message

The session code is the one reported through the connection registration. Session code zero applies to the LNA-MAU session.

The session control message is one of those described in 5.2.4. The SESSION_DOWN code shall be ignored when it is sent from a MAU, except when it is used on session zero.

8.2.3 Interface definition messages

8.2.3.1 Request interface open (***MAPI_IREQ***)

This message is used by the MAU to define an interface. If the interface is specified as accept type, the LNA will add accept type data objects to the system. If the interface is specified as connect type, the LNA will try to connect this interface to the corresponding interface in the system.

Several interface definition messages may be issued by both client and server for the definition of one interface. For both accept and connect interfaces, all MCPs must be accepted for the interface to be activated.

Table 21 – Request interface open

Octet #	Data type	Field description
0 – 1	word16_m	MV4_MAGIC
2 – 3	word16_m	MAPI_IREQ
4 – 5	word16_m	Priority
6 – 7	word16_m	Length = x + 1
8 – 9	mcid_m	Interface identity
10 – 11	word16_m	Session (not used)
12 – 15	word32_m	Transaction code (not used)

Octet #	Data type	Field description	
16 – 19	word32_m	Timeout (ms)	
20 – 23	int32_m	IF type (M_CONNECT or M_ACCEPT)	
24 – 25	word16_m	Maximum pending transactions (used only for accept)	
26 – 27	word16_m	Maximum number of clients (used only for accept)	
28 – 28	bool_m	More flag (true if more MCPs follow later)	
29 –	mauname_m	MAU name (null terminated, may be null for accept)	
-	ifname_m	IF name (null terminated)	
-	password_m	IF password (null terminated, may be null for accept)	
-	word16_m	Number of MCPs in this message	
-	mcid_m	ID of new MCP	Information repeats for each MCP.
-	mcpname_m	MCP name (null terminated)	
- x	format_m	Format string (null terminated)	

The interface identity code is in another scope than that of the MCP identity codes so the same numeric values can be used for MCP and IF codes.

The maximum pending transactions and maximum clients are used on the server side to control MAU load. The limits are enforced by the LNA. The value zero in these fields means no limit to be enforced.

If a non-zero connection limit is specified, the server LNA shall deny the connection attempt immediately, if the requesting MAU has not already established a session with the server MAU.

NOTE This should be used together with the authentication companion standard IEC 61162-420 to avoid that non-authenticated client MAUs block a service interface.

The interface identity, the load limit parameters, the interface and MAU names must be the same in all messages if more are needed to define a complete interface. The more flag shall be zero in all messages except the last, if more messages are used to define one interface. The server MAU shall either use its own name or null in the MAU name field.

The server MAU should leave the password field empty. The connection request message will contain the password the client specified. The LNA will not check passwords.

The MCP identity must be unique on a MAU. The MCP name and format string are discussed in clause 9. The MCP specific information is repeated for every MCP in the message.

Note that anonymous broadcasts can use any MAU name. The interface name is used to distinguish between the available broadcast ports. Only ABC MCPs can be defined in an interface associated with an ABC port.

8.2.3.2 Acknowledge interface open (*MAPI_IACK*)

This message is used to report the result of the LNA's attempt to define MCPs in an interface. The status code is LM_OK for a successful attempt or one of the following error codes:

a) LM_TIMEOUT: the request timed out, opening aborted;
b) LM_FORMAT_FAIL: the format of the request was wrong (bad interface parameters);
c) LM_INVALID_MCP: the request contained a useless MCP identity code;
d) LM_IF_EXISTS: the interface already exists;
e) LM_PASSWORD_FAIL: the remote interface was found, but the passwords did not match;

f) LM_AUTHEN_FAIL: authentication failed at server MAU;

g) LM_CLIENTS: too many clients.

Table 22 – Interface open acknowledge

Octet #	Data type	Field description	
0 – 1	word16_m	MV4_MAGIC	
2 – 3	word16_m	MAPI_IACK	
4 – 5	word16_m	Priority	
6 – 7	word16_m	Length = x + 1	
8 – 9	mcid_m	Identity of IF (as defined by request)	
10 – 11	word16_m	Session (for connect)	
12 – 15	word32_m	Transaction identity (not used)	
16 – 19	word32_m	General status	
20 – 21	word16_m	Number of bad MCPs	
22 –	mcid_m	MCP identity	Repeat for all bad MCPs
- x	word32_m	Status	

The LNA sends one acknowledgement for all MCPs in the interface, if the attempt succeeded. The status code will be LM_OK in that case and the MCPs field will be zero. If an error occurred, the general error code will be put into the status field and MCP specific errors (if appropriate) will follow for the MCPs that are flagged as bad. It is up to the LNAs to put as much information as appropriate into this message. The list of bad MCPs need not be complete, for example if the message length limits the number. It is only meant as a diagnostic help to the system integrator.

The interface identity code is the code used in the request.

The session code is used at the client side to identify the remote MAU session. Later connections or transactions belonging to the same session will use the same session code.

8.2.3.3 Interface connection request to server (*MAPI_IOPEN*)

This message is sent to the server to indicate that a client attempts to connect to the server's interface. Only one message is sent regardless of the number of connection request messages the client sends. The MAU is not notified of the MCPs to which the client has connected. The message shall be responded to by a MAPI_IOACK message.

Table 23 – Interface connect request to server

Octet #	Data type	Field description
0 – 1	word16_m	MV4_MAGIC
2 – 3	word16_m	MAPI_IOPEN
4 – 5	word16_m	Priority
6 – 7	word16_m	Length = x + 1
8 – 9	mcid_m	Identity of IF (as defined by interface definition request)
10 – 11	word16_m	Session code
12 – 15	word32_m	Transaction identity (not used)
16 – 19	word32_m	Status/timeout (not used)
20 –	mauname_m	Remote MAU name (null terminated)
- x	password_m	Remote MAU's request password (null terminated)

The identity field shall be the one the server MAU defined for this interface. The session field identifies the remote MAU session and will be used in all transactions belonging to this session. The remote MAU name and password is as specified in the connection attempt by the client.

8.2.3.4 Interface connect acknowledge from server (***MAPI_IOACK***)

This message is sent by the server to reply to a client's connection attempt. The status field gives the status of the request:

a) `LM_OK`: authentication given;

b) `LM_PASSWORD_FAIL`: passwords did not match;

c) `LM_AUTHEN_FAIL`: authentication failed for other reasons.

Table 24 – Interface connect acknowledge from server

Octet #	Data type	Field description
0 – 1	word16_m	MV4_MAGIC
2 – 3	word16_m	MAPI_IOACK
4 – 5	word16_m	Priority
6 – 7	word16_m	Length = 20
8 – 9	mcid_m	Identity of IF (as defined by interface definition request)
10 – 11	word16_m	Session code (as in request)
12 – 15	word32_m	Transaction identity (not used)
16 – 19	word32_m	Status of request

The identity field shall be that the server MAU defined for this interface. The session field is the same as in the request.

8.2.3.5 Interface remove (***MAPI_IREM***)

This message is used by a server or client to remove a complete interface (connection).

Table 25 – Interface remove

Octet #	Data type	Field description
0 – 1	word16_m	MV4_MAGIC
2 – 3	word16_m	MAPI_IREM
4 – 5	word16_m	Priority
6 – 7	word16_m	Length = 20
8 – 9	mcid_m	Identity of IF (as defined by interface definition request)
10 – 11	word16_m	Session (not used)
12 – 15	word32_m	Transaction identity (not used)
16 – 19	word32_m	Timeout (not used)

8.2.3.6 Interface connection down (***MAPI_IDOWN***)

This message is sent to a server MAU when one client disconnects from an interface. Note that the general session down message (8.2.2.4) can have the same effect as this message. No interface down messages are sent if a remote session is terminated.

Table 26 – Interface connection down

Octet #	Data type	Field description
0 – 1	word16_m	MV4_MAGIC
2 – 3	word16_m	MAPI_IDOWN
4 – 5	word16_m	Priority
6 – 7	word16_m	Length = 20
8 – 9	mcid_m	Identity of IF (as defined by interface definition request)
10 – 11	word16_m	Session
12 – 15	word32_m	Transaction identity (not used)
16 – 19	word32_m	Timeout (not used)

The session code is the one reported through the connection registration.

The server MAU must cancel any pending transaction requests belonging to this session on this interface when this message is received. It may also delete a client MAU's authentication for accessing other interfaces.

8.2.4 Data transfer messages

8.2.4.1 Transaction request messages (*MAPI_xREQ*)

The client MAU can request data from a server MAU by using the message format described in Table 27. The same message is also used to send data to servers. The LNA passes this message encapsulated in a LL_DATA message to the remote LNA. The reply, if any, will be delivered to the sending MAU as a MAPI_xACK message. The x represents a letter code corresponding to the data object type as listed in Table 13.

Table 27 – Transaction request message

Octet #	Data type	Field description
0 – 1	word16_m	MV4_MAGIC
2 – 3	word16_m	*type:* MAPI_xREQ
4 – 5	word16_m	Priority
6 – 7	word16_m	Length = 20 or x+1
8 – 9	mcid_m	*ident*, ID of MCP
10 – 11	word16_m	Session (only for server)
12 – 15	word32_m	Transaction code (to be returned in acknowledgement)
16 – 19	word32_m	Timeout (client MAU only)
20 – x	word8_m[]	*Data*

The same request message will be used on the client and the server side. The message will have the same contents except for the *ident* field, which may change value when it is passed through the LNAs.

The client's LNA shall change the client MAU's MCP ID to the MCP ID of the server MAU.

The session code shall be set by the server's LNA to the code used in interface connection establishment (where applicable).

The transaction code is set by one sending entity and this entity requires that the same code shall be returned in the corresponding acknowledgement (where applicable). It is the receiver's responsibility (LNA or MAU) to make sure that the return transaction code is correct.

The timeout is used by the client MAU and is enforced by the local LNA. No other entity shall use the timeout code.

The request message can contain data when it is marked with a 'W' in the Read/Write column of.Table 13. It is, however, legal to have an empty data field even for a write type message. The *data* representation mechanisms are described in 3.4.

8.2.4.2 Transaction acknowledge messages (*MAPI_xACK*)

The transaction request message will normally be acknowledged by the server MAU (except for non-acknowledged write transfers and ordinary subscriptions that are copied by the remote LNA). The format of this message is shown in Table 28. The acknowledge can also be generated by one of the intermediate LNAs if an error occurs during the transfer. A positive acknowledge will contain the status code LM_OK. Any other code signals an error:

a) LM_INVALID_MCP: the local LNA could not find the specified MCP;
b) LM_TIMEOUT: the local LNA has detected that the request has timed out according to the given value in the request;
c) LM_NO_SUCH_MCP: the remote LNA can not find the specified remote MCP;
d) LM_FORMAT_FAIL: the remote LNA detects non-conformity between request type and format of the MCP;
e) LM_CONGESTION: message dropped due to congestion;
f) LM_LIMIT: message dropped due to a transaction number limit;
g) LM_AUTHEN_FAIL: MAU denied service due to authentication problems.

Table 28 – Transaction acknowledge message

Octet #	Data type	Field description
0 – 1	word16_m	MV4_MAGIC
2 – 3	word16_m	*type*: MAPI_xACK
4 – 5	word16_m	Priority
6 – 7	word16_m	Length = 20 or x+1
8 – 9	mcid_m	*ident*, ID of MCP
10 – 11	word16_m	Session (used by client)
12 – 15	word32_m	Transaction code (as in request)
16 – 19	word32_m	*status:* LM_OK or error
20 – x	word8_m[]	User defined data (only if LM_OK)

The same acknowledge message will be used on the client and the server side. The message will have the same contents except for the *ident* field and possibly the transaction code fields, which may change value when it is passed through the LNAs. The server's LNA shall change the server MAU's MCP ID to the MCP ID of the client MAU. The transaction code shall be changed by all entities to the code of the corresponding request.

The acknowledge message can contain data when it is marked with an 'R' in the Read/Write column of Table 13. It is, however, legal to have an empty data field even for a read type message. The data representation mechanisms are described in 3.4.

The LNA shall check that an acknowledgement from another LNA (encapsulated in a LL_DATA message) corresponds to a local pending request.

8.2.4.3 Special considerations for subscribe type acknowledge.

The server LNA maintains the list of remote MCPs that are subscribing to a given server MAU's subscribe type data object. The following notes apply (see Table 15):

a) a MAPI_FxACK type acknowledge shall only be sent to the requesting MCP;

b) a MAPI_SACK acknowledge shall be sent to all subscribing MCPs;

c) a MAPI_BACK or MAPI_AACK acknowledge shall be sent once via multi-cast;

d) a MAPI_DACK shall be sent only to the MAU identified by the transaction code;

e) the receipt of an LL_NOMCP message relating to a subscribing MCP shall, if possible, remove the appropriate transaction object from the list of subscribers. The same applies to the LL_DATAC and all session or interface close messages;

f) the server MAU may issue MAPI_SACK messages even though no MCPs are subscribing. These messages shall be discarded by the LNA.

8.2.4.4 Transaction cancel request (*MAPI_CREQ*)

The MAU can cancel an outstanding transaction request by using the cancel request message. The message format is described in Table 29. It shall be acknowledged by a MAPI_CACK message.

Table 29 – Transaction cancel request message

Octet #	Data type	Field description
0 – 1	word16_m	MV4_MAGIC
2 – 3	word16_m	MAPI_CREQ
4 – 5	word16_m	Priority
6 – 7	word16_m	Length = 20
8 – 9	mcid_m	ID of MCP
10 – 11	word16_m	Session (only for server)
12 – 15	word32_m	Transaction code
16 – 19	word32_m	Timeout (ignored)

The session code shall be set for the server MAU. The transaction code is that used in the corresponding request message (possibly exchanged by LNAs).

8.2.4.5 Transaction cancel acknowledge (*MAPI_CACK*).

The MAU's cancel request message shall always be acknowledged with a cancel acknowledge message. The message format is shown in Table 30. The status code is LM_OK if the LNA was able to detect a pending data message. Other status codes that may be returned are:

a) LM_NOTHING_TO_CANCEL: no acknowledgement pending for this transaction;

b) LM_INVALID_MCP: the MCP was not found on the client's LNA.

Table 30 – Transaction cancel acknowledge message

Octet #	Data type	Field description
0 – 1	`word16_m`	`MV4_MAGIC`
2 – 3	`word16_m`	`MAPI_CACK`
4 – 5	`word16_m`	Priority
6 – 7	`word16_m`	Length = 20
8 – 9	`mcid_m`	ID of MCP
10 – 11	`word16_m`	Session (for server)
12 – 15	`word32_m`	Transaction code
16 – 19	`word32_m`	*status:* `LM_OK` or error code.

The session code shall be the one applicable to the server MAU. The transaction code shall be the one specified for the request to be cancelled.

8.3 LNA-LNA message formats for reliable link

The reliable link is used for all control messages between LNAs and for all reliable data transfers. The broadcast link is used for broadcast data.

8.3.1 General message format

All messages between LNAs use the general message format described in 8.1.1. All provisions of that subclause apply to the LNA-LNA messages.

8.3.2 Connection management

8.3.2.1 Connection establishment

Connection establishment between LNAs is handled by the T-profile. The TLI will report the success of a connection request or the incoming connection attempt by referencing it to the network node number (NNN) of the remote host computer and the TP network at which the connection was registered. The T-profile will make sure that the reported link connection point is the only one used for communication with that remote LNA.

8.3.2.2 Connection watchdog (*LL_ALIVE*)

This message is used by an LNA to check or verify if an LNA-LNA link is alive.

Table 31 – Connection watchdog message

Octet #	Data type	Field description
0 – 1	`word16_m`	`MV4_MAGIC`
2 – 3	`word16_m`	`LL_ALIVE`
4 – 5	`word16_m`	Priority
6 – 7	`word16_m`	Length = 10
8 – 9	`word16_m`	Status: 0 = request answer

The status code is used to ask for an answer. Zero in this field shall cause the receiver to immediately reply with a similar message. The answer should normally have the status field set to non-zero to avoid looping.

Failing to answer to this message within a deadline defined in 6.3.4 will cause the requesting LNA to close the communication link and flag the LNA as dead.

8.3.3 MAU management

8.3.3.1 MAU request message (*LL_MAUREQ*)

This message is used to request the confirmation of a specified MAU's location. The message format is described in Table 32. A positive or negative acknowledge is sent in a LL_MAUACK message.

Table 32 – MAU request message

Octet #	Data type	Field description
0 – 1	word16_m	MV4_MAGIC
2 – 3	word16_m	LL_MAUREQ
4 – 5	word16_m	Priority
6 – 7	word16_m	Length = x+1
8 – 9	mauid_m	Requestor – MAU id
10 – x	mauname_m	MAU name string

The MAU identity code transmitted is the identity code allocated by the requesting LNA for this MAU (corresponding to the LNA's MAU database object described in Figure 10). This identity code is also used in the acknowledgement message (see next subclause).

8.3.3.2 MAU acknowledge message (*LL_MAUACK*)

This message is used to confirm a requested MAU's location or to report the death of a connected MAU to remote users. The message format is described in Table 33. The LNA that receives an LL_MAUREQ message shall check if the specified MAU exists locally and format the acknowledge message accordingly.

Table 33 – MAU acknowledge message

Octet #	Data type	Field description
0 – 1	word16_m	MV4_MAGIC
2 – 3	word16_m	LL_MAUACK
4 – 5	word16_m	Priority
6 – 7	word16_m	Length = 12 or 14
8 – 9	mauid_m	Requestor MAU id (from request message)
10 – 11	word16_m	Found
12 – 13	mauid_m	Local MAU id (if MAU is known)

The LL_MAUACK message contains the identity reported by the requesting LNA and then the result of the name look-up in the *found* field:

a) LM_OK (zero) is used if the MAU was found as a result of a previous request, the identity code follows;

b) LM_RMAU_DOWN is used if a defined MAU has died, the identity code follows;

c) LM_MAU_NOT_FOUND is used if the MAU was not found as a result of a request. No identity code follows.

The receiver of this message, when the remote MAU is signalled down, shall inform all its MAUs that have connections to the dead MAU about its death. Internal LNA states shall be cleaned up and MCP reconnection attempts shall be started.

8.3.3.3 Congestion control message (*LL_SESSION*)

This message is used to report the congestion state for a specific MAU. It is sent by an LNA as a result of a request from one of its MAUs. The message can be directed to a specific remote MAU or be a general congestion report for all messages to the afflicted MAU.

Table 34 – Congestion control message

Octet #	Data type	Field description
0 – 1	word16_m	MV4_MAGIC
2 – 3	word16_m	LL_SESSION
4 – 5	word16_m	Priority
6 – 7	word16_m	Length = 16
8 – 9	mauid_m	Afflicted MAU on sender LNA
10 – 11	mauid_m	Offending MAU on receiving LNA (or zero)
12 – 15	word32_m	Congestion control message

The afflicted MAU shall always be specified. The offending MAU is only specified when a specific MAU-MAU association is addressed. If the offending MAU code is zero, all MAUs on the receiving LNA that have connections to the afflicted MAU shall get the congestion control message.

The congestion control messages are defined in 5.2.4. The SESSION_DOWN code shall be ignored.

8.3.3.4 MAU or MCP unknown (*LL_NOMCP*)

This message is used to report an unknown MAU or MCP identity as a result of a previous request. The receiver of this message shall check internal state and try to determine the cause of the message. However, the message may be caused, for example by crossing cancellations and data acknowledgements, and may not always be traceable to a bad state.

Table 35 – Unknown MAU or MCP

Octet #	Data type	Field description
0 – 1	word16_m	MV4_MAGIC
2 – 3	word16_m	LL_NOMCP
4 – 5	word16_m	Priority
6 – 7	word16_m	Length = 18
8 – 9	word16_m	Offending message/request
10 – 11	mauid_m	MAU id (as transmitted in request – or zero)
12 – 13	mcid_m	MCP id (as transmitted in request – or zero)
14 – 17	word32_m	Transaction id (as transmitted in request – or zero)

The offending message field contains the message code for the offending request (typically LL_DATA or LL_DATAC). The rest of the message contains target information as used in the offending LNA's message identification codes. Fields that are not relevant are zero.

8.3.4 Interface connection management

8.3.4.1 Interface open request message (*LL_IFREQ*)

This message is used to request the opening of a connection to a server's interface on a remote LNA. The message is compiled by the sending LNA from MAPI_IREQ requests from its local MAU.

Table 36 – IF open request message

Octet #	Data type	Field description	
0 – 1	word16_m	MV4_MAGIC	
2 – 3	word16_m	LL_IFREQ	
4 – 5	word16_m	Priority	
6 – 7	word16_m	Length = x+1	
8 – 9	mauid_m	ID of connecting MAU	
10 – 11	mcid_m	ID of connecting IF	
12 – 12	bool_m	More (true if more messages follow)	
13 –	mauname_m	Server MAU name (null terminated)	
–	ifname_m	IF name (null terminated)	
–	password_m	IF password (null terminated)	
–	word16_m	Number of MCPs in this message	
–	mcid_m	ID of new MCP	Information repeats for each MCP
–	mcpname_m	MCP name (null terminated)	
– x	format_m	Format string (null terminated)	

The data entries and the formats are the same as in the MAPI_IREQ message.

NOTE In particular, the main parts of the message are identical in layout to the MAU-LNA message. This can be utilized to make processing more efficient.

8.3.4.2 IF acknowledge message(*LL_IFACK*)

This message is used by a server MAU's LNA to acknowledge the connection of a number of MCPs in an interface.

Table 37 – Interface open acknowledge

Octet #	Data type	Field description	
0 – 1	word16_m	MV4_MAGIC	
2 – 3	word16_m	LL_IFACK	
4 – 5	word16_m	Priority	
6 – 7	word16_m	Length = x + 1	
8 – 9	mauid_m	ID of connecting MAU	
10 – 11	mcid_m	ID of connecting IF (as specified in request)	
12 – 13	mauid_m	ID of accepting MAU	
14 – 15	mcid_m	ID of accepted IF	
16 – 16	bool_m	*More (true if more messages follow)*	
17 – 18	word16_m	*Result*: LM_OK or error code	
19 – 20	word16_m	MCPs in this message	
21 –	mcid_m	MCP identity	Either repeat for all bad MCPs
- x	word32_m	Error code	
21 –	mcid_m	Client MCP identity	or repeat for all MCPs
- x	mcid_m	Server MCP identity	

The server LNA reports back either all mappings between client and server MCP codes or a list of bad MCPs, if applicable. The mapping is only returned when the general message status is connection established. The error codes are only returned when the general message status is connection denied and there were errors related to individual MCPs. The server LNA does not have to return any MCPs, but it is suggested that error codes are returned whenever possible to ease diagnostics.

An unsuccessful match is indicated with one of the following codes:

a) LM_RMAU_DOWN: the MAU was not at this LNA, all fields invalid;
b) LM_NO_INTERFACE: the interface was not found, all fields invalid;
c) LM_INVALID_MCP–: the MAU and interface were found, but an error in matching MCP occurred. Bad MCPs follow;
d) LM_FORMAT_FAIL: the format string was wrong. Bad MCPs follow;
e) LM_PASSWORD_FAIL: both MAU and interface were found, but the passwords did not match, only the accepting MAU identity field is valid. The connection was denied by the server MAU;
f) LM_AUTHEN_FAIL: authentication failed at server MAU (for reason other than password mismatch). The connection was denied by the server MAU;
g) LM_CLIENTS: too many clients.

All requested MCPs must be matched to establish a connection. All identity mapping codes must be returned. The server LNA does not have to store client identity codes. It is the client's responsibility to supply the correct identity codes for the remote MCP.

8.3.4.3 Removed interface message (*LL_IFREM*)

This message is used to inform a remote LNA that the MAU's interface (accept type) or connection to an interface (connect type) no longer is available for data transactions. The receiving LNA shall remove all states related to the removed interface.

Table 38 – Removed interface message

Octet #	Data type	Field description
0 – 1	word16_m	MV4_MAGIC
2 – 3	word16_m	LL_IFREM
4 – 5	word16_m	Priority
6 – 7	word16_m	Length = 16
8 – 9	mauid_m	ID of sender MAU
10 – 11	mcid_m	ID of sender interface
12 – 13	mauid_m	ID of receiver MAU
14 – 15	mcid_m	ID of receiver interface

Errors in message formats shall be ignored, but reported on a diagnostic console if possible.

8.3.5 Data transfer management

8.3.5.1 Data transfer message (LL_DATA)

The data transfer message is used to transport MAU data messages (MAPI_xREQ and MAPI_xACK) when in transit between LNAs. The message format is shown in the table below.

Table 39 – Data transfer message

Octet #	Data type	Field description
0 – 1	word16_m	MV4_MAGIC
2 – 3	word16_m	LL_DATA
4 – 5	word16_m	Priority
6 – 7	word16_m	Length = x+1
8 – 9	mauid_m	ID of receiver MAU
10 – 11	mcid_m	ID of receiver MCP
12 – 13	mauid_m	ID of sender MAU
14 – 15	mcid_m	ID of sender MCP
16 – x	word8_m[]	MAU message including header

The inclusion of MAU and MCP codes can be used to verify the correctness of the message. This information shall also be available from the general MAU and MCP look-up process.

The MAU data transfer message is always included verbatim in the body of this message. The LNA should try to minimize copying by using this fact.

Before a message received from a MAU is sent to a remote LNA, the local LNA needs to modify the MAU message as follows:

a) change transaction code if necessary. The transaction code shall be unique for the LNA (within the precision of the code) or it shall correspond to the transaction request to which this acknowledgement belongs. Discard messages that have wrong transaction codes (the transaction was most likely cancelled);
b) change target MCP ID to that of the remote MAU;
c) set priority of LNA message according to that of the MAU message.

Before the MAU message can be sent to the MAU, the LNA needs to modify the message as follows:

a) check incoming transaction code (for acknowledgements) against request transaction code. Discard messages with wrong code (the request was probably cancelled);
b) change transaction code, if necessary, to that of the corresponding request.

8.3.5.2 Data cancel message (*LL_DATAC*)

The data transfer message is used to transport MAU data cancel messages (MAPI_CREQ and MAPI_CACK) when in transit between LNAs. The message format is shown in Table 40.

Table 40 – Data cancel transfer message

Octet #	Data type	Field description
0 – 1	word16_m	MV4_MAGIC
2 – 3	word16_m	LL_DATAC
4 – 5	word16_m	Priority
6 – 7	word16_m	Length = x+1
8 – 9	mauid_m	ID of receiver MAU
10 – 11	mcid_m	ID of receiver MCP
12 – 13	mauid_m	ID of sender MAU
14 – 15	mcid_m	ID of sender MCP
16 – x	word8_m[]	MAU message including header

Note that the message format is identical to that of the LL_DATA message. See subclause 8.3.5.1 for comments on processing and header data. The same comments regarding processing apply also to this message.

8.4 LNA-LNA message formats for multicast link

The multi-cast link is used for all name look-up, watchdog activities and general broadcast of data.

8.4.1 General message format

All multi-cast messages between LNAs use the general message format described in 8.1.1. All provisions of that subclause apply to these messages.

Note that multi-cast messages typically have limited message length and that special restrictions on message contents can occur.

8.4.2 Different multi-cast ports

All LNAs shall maintain a number of listening ports for LNA-LNA multicast messages. The following groups of ports are defined:

a) ABC0: standard port for LNA related broadcasts (MAU name requests and acknowledgements and general broadcast subscriptions);
b) ABC1 to ABC5: ports for anonymous broadcast messages.

NOTE The client and server LNAs must check the format string to determine if a subscribe type MCP shall use the broadcast or the reliable port. It also needs to examine the MAU name to see which anonymous port to use. The messages from the MAU are identical for all types of subscribe.

The T-profile is required to establish these listening ports depending upon the properties of the TP network in use. This includes subscribing to multi-cast messages if appropriate.

8.4.3 Name look-up and watchdog messages

8.4.3.1 Request MAU location (*CC_REQMAU*)

The LNA can use the message shown in Table 41 to request the location of a MAU. The message is answered by the LNA that serves the MAU with a CC_DEFMAU message.

Table 41 – Request MAU location

Octet #	Data type	Field description
0 – 1	word16_m	MV4_MAGIC
2 – 3	word16_m	CC_REQMAU
4 – 5	word16_m	Priority
6 – 7	word16_m	Length = x+1
8 – x	mauname_m	MAU name string

This message shall only be used in the first requests for a MAU. Later requests shall normally use CC_WATCHDOG.

8.4.3.2 Define MAU location (*CC_DEFMAU*)

The message shown in Table 42 is used to broadcast the location of a MAU. It is only sent by the LNA that serves this MAU. The message shall be sent when the MAU is defined by the LNA and each time the MAU's location is requested through a CC_REQMAU or a CC_WATCHDOG message.

Table 42 – Report MAU location

Octet #	Data type	Field description
0 – 1	word16_m	MV4_MAGIC
2 – 3	word16_m	CC_DEFMAU
4 – 5	word16_m	Priority
6 – 7	word16_m	Length = x+1
8 – 11	word32_m	Network node number of LNA serving MAU
12 – 13	mauid_m	MAU identity code
14 – x	mauname_m	MAU name

8.4.3.3 Warning about duplicate MAU (*CC_METOO*)

The message described in Table 43 is used by LNAs that have a local MAU in their database with a name equivalent to one reported by a CC_DEFMAU or CC_WATCHDOG message.

Table 43 – Report duplicate MAU

Octet #	Data type	Field description
0 – 1	word16_m	MV4_MAGIC
2 – 3	word16_m	CC_METOO
4 – 5	word16_m	Priority
6 – 7	word16_m	Length = x+1
8 – 11	word32_m	Network node number of LNA serving MAU
12 – 13	mauid_m	MAU identity code
14 – x	mauname_m	MAU name string

This message is currently used only to inform about a potential configuration problem in the system. An LNA receiving this message shall report it as a system problem.

8.4.3.4 Report dead MAU (*CC_DEADMAU*.

The message described in Table 44 is used to report that a MAU local to an LNA has died. LNAs receiving this message shall remove that MAU from their name look-up databases.

Table 44 – Report dead MAU

Octet #	Data type	Field description
0 – 1	word16_m	MV4_MAGIC
2 – 3	word16_m	CC_DEADMAU
4 – 5	word16_m	Priority
6 – 7	word16_m	Length = x + 1
8 – 11	word32_m	Network node number of LNA serving MAU
12 – 13	mauid_m	MAU identity code
14 – x	mauname_m	MAU name string

8.4.3.5 LNA watchdog message (*CC_WATCHDOG*).

The message described in Table 45 is used to send regular LNA watchdog messages, including information about known and unknown MAUs.

Table 45 – LNA watchdog

Octet #	Data type	Field description
0 – 1	word16_m	MV4_MAGIC
2 – 3	word16_m	CC_WATCHDOG
4 – 5	word16_m	Priority
6 – 7	word16_m	Length = x + 1
8 – 11	word32_m	Network node number of LNA
12 – 13	word16_m	Requested MAU names
14 – 15	word16_m	Known MAU names
16 – x	[]mauname_m	MAU name strings

The LNA shall give priority to sending requested MAUs, but should also try to send as many known MAUs as possible, given the finite length of the broadcast message. The order of MAU names are requested MAUs first and known MAUs after. The total of the MAU name counts and messages shall be equal.

8.4.4 Data transfer messages

8.4.4.1 Ordinary broadcast data transfer message (*LLBC_SACK*)

This message contains an ordinary unreliable subscribe, server initiated data acknowledge package. The message format is described in Table 46.

Table 46 – Ordinary broadcast data transfer message

Octet #	Data type	Field description
0 – 1	`word16_m`	`MV4_MAGIC`
2 – 3	`word16_m`	`LLBC_SACK`
4 – 5	`word16_m`	Priority
6 – 7	`word16_m`	Length = x + 1
8 – 9	`mauid_m`	ID of server MAU
10 – 11	`mcid_m`	ID of server MCP
12 – x	`word8_m[]`	MAU message including header (`MAPI_BACK` type)

The receiving LNA shall distribute the embedded MAU message to the subscribing client MCP or MCPs local to the LNA. For this purpose, the LNA needs to maintain a list over subscribing MCPs. The server's identity codes must be used to distinguish between different broadcast subscribe sources. All broadcast messages will normally arrive at the same network port (dependent on transport protocol in use – see IEC 61162-410).

The MAU message needs to be modified by the receiving LNA to change MCP identity codes to the code appropriate for the receiving MAU (see 8.2.3.3 and 8.2.4.2).

8.4.4.2 Anonymous broadcast type data transfer message (*LLBC_AACK*)

This message contains an anonymous unreliable subscribe data acknowledge package. The message format is described in the table below.

Table 47 – Anonymous broadcast data transfer message

Octet #	Data type	Field description
0 – 1	`word16_m`	`MV4_MAGIC`
2 – 3	`word16_m`	`LLBC_AACK`
4 – 5	`word16_m`	Priority
6 – 7	`word16_m`	Length = x + 1
8 – 15	`[8]char8_m`	MPC name (8 left-most characters, no null)
16 – 19	`word32_m`	Format hash
20 – 21	`int16_m`	Format string length
22 – x	`[]word8_m`	MAU message including header (`MAPI_AACK` type)

See previous clause for details on processing. See also 8.4.5 for details on calculation of MCP attribute values.

The MAU message needs to be modified by the receiving LNA to change MCP identity codes to the code appropriate for the receiving MAU (see 8.2.3.3 and 8.2.4.2).

8.4.5 Anonymous broadcast message details

8.4.5.1 General

The data message associated with anonymous broadcasts must embed some parts of the format string and MCP names in the message itself. Otherwise, it will not be possible to distinguish different messages (from different MCPs) arriving at the same ABC port.

8.4.5.2 Encoding of MAU name

There is no MAU associated with anonymous broadcasts.

8.4.5.3 Encoding of interface and MCP name

The MCPs supporting anonymous broadcast must use the interface corresponding to the appropriate ABC port. The MCP name is restricted to eight characters which are embedded in the message. The interface is encoded implicitly in the ABC port used. The eight left-most characters (first characters) of the MCP name are used. No null padding shall be used if all characters are transmitted, otherwise, all non-used characters shall be null.

8.4.5.4 Encoding of format string

The format string is encoded in six octets. Two octets (as one `int16_m`) give the length of the format string including leading "`a/`", but excluding terminating null. Four octets (`word32_m`) give a hash value of the format string derived as described below.

a) Strip the format string of the leading two characters: "`a/`".
b) Null pad the string at the end until it is an integral number of four octets long.
c) Divide the string into groups of four consecutive octets. Index the elements from zero to three from the string start towards the end.
d) Create four sums of octets, each a sum of all group elements with the same index (without carry from one octet to the next).
e) Assemble the resultant four octets into a `word32_m` by using index zero as the most significant octet, index one as the next and so on.
f) Use the resulting `word32_m` as the hash value.

8.4.5.5 Processing of messages

The LNA that processes a client side connection request shall calculate the format string length and the hash value and use these to check incoming packets for matches to the connection type MCPs, together with the MCP name.

The server LNA shall likewise calculate the two values and insert them into the outgoing message together with the MCP name.

8.4.5.6 Example of the format string

The format string "a/(w32[32]c8f32f32)" has the octet breakdown shown in the table below.

Table 48 – Example ABC format calculation

Character	Hex value	Sum 0	Sum 1	Sum 2	Sum 3
a/	61,2f				
(w32	28,77,33,32	28	77	33	32
[32]	5b,33,32,5d	83	AA	65	8F
c8f3	63,38,66,33	E6	E2	CB	C2
2f32	32,66,33,32	18	48	FE	F4
)	29,0,0,0	41	48	FE	F4

The resulting format hash number will be 0x4148fef4. The length will be 19 octets.

9 General identity codes

9.1 Protocol and software version codes

The version code consists of a major (to the left of the decimal) and a minor (to the right) number. The format is N.M, where N and M are two positive decimal numbers. M can be zero.

The minor number shall be incremented when a modification is made that has no or little significance for functionality. It is assumed that these modifications will retain full compatibility with older versions of the protocol.

The major number shall be incremented when a significant change is made in the protocol. Significant means that older versions cannot be fully compatible with the new. Care shall, however, be taken to achieve upward compatibility, i.e. that older nodes can continue to use old functionality together with new nodes.

Sometimes a third number is used, giving a patch level for the current version. The third number will normally be placed after a decimal sign at the right of the two other numbers.

9.2 Network address, node number and LNA id (`address_m`, `word32_m`)

The T-profile address-specification is of the type `address_m`. The format of the address is the same for all nodes (LNAs) on one T-profile network (TP-network). Each TP-network shall have one network address that is the same for all nodes and identifies the network on which the nodes are located.

The TP-address specification is composed of a 16 bit unsigned integer specifying the type of protocol and a variable length array of up to 48 octets. The contents of the array will depend on the type of address. The address types and array interpretation are determined by the T-profiles and the companion standard IEC 61162-420.

Within the TP-network, the individual LNAs are identified with the network node number – NNN (`word32_m`). The LNA identity is defined as the network node number on a given TP-network.

9.3 MAU identity (mauname_m, mauid_m)

A MAU (connected or disconnected) shall be assigned a symbolic name (`mauname_m`). The MAU name is a null terminated character string with a length of maximum `MAUNAME_LEN` characters excluding null. Any character except null and a leading underscore (‘_’) is allowed in the string.

NOTE The standard allows a server MAU to use a leading underscore ('_') in a dummy name to inform the LNA that it should select a name for the MAU. This means that it is impossible for a server MAU to use a leading underscore in its configured name.

The identity of a MAU connected to the network is determined by the MAU identity code (`mauid_m`) together with its LNA's identity code (network address). The standard does not guarantee that two MAUs with identical symbolic names do not connect to the network. This means that the (`address_m`, `mauid_m`) tuple is the only unambiguous identification.

The following MAU names are reserved by this standard:

a) _.*: a leading underscore is not allowed as a general MAU name (see above);

b) LNAhhhhhhhh: the LNA associated built-in MAU. The string following the LNA is the network node number of the LNA encoded in hexadecimal (most significant octet first), using only lower case letters and leading zeros.

9.4 Data object and MCP identity (`mcid_m`)

There are three forms of data object related identity codes. These are schematically illustrated as heavy outlined and numbered (1 to 3) rectangles in Figure 32.

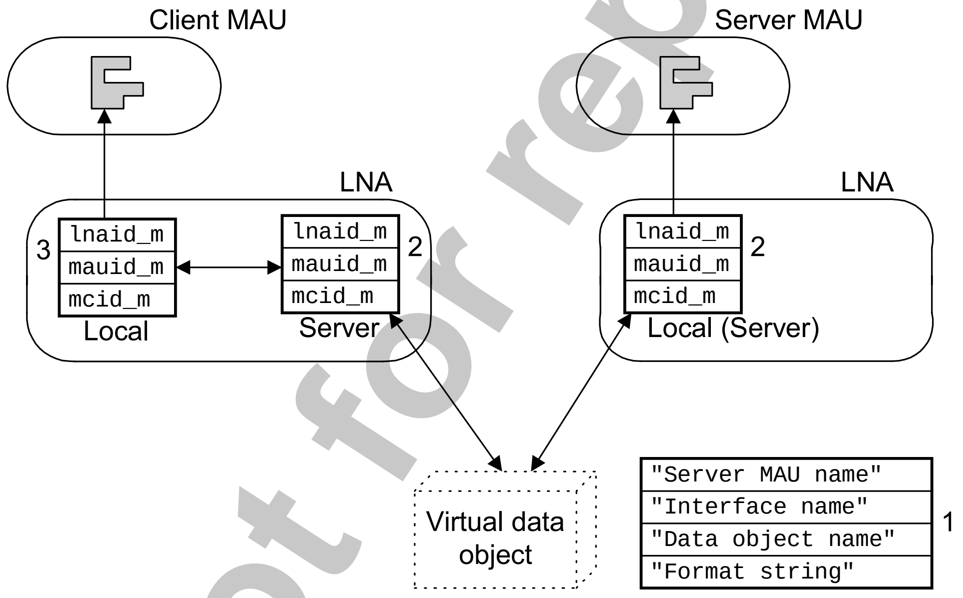

Figure 32 – Variants of data object identity

The identity forms are:

a) when a server MAU has defined a new MCP one can look at this as a virtual data object existing somewhere in the network. This object is identified by a set of attribute values in the form of symbolic text strings:
   - server MAU name,
   - interface name,
   - data object name,
   - format string containing transaction type and data record structure;

b) the server MAU uses a numerical **MCP** (MAU connection point) code (`mcid_m`) internally to reference the data object. This code together with the identity codes for the server MAU and the LNA is a global identification triplet used by the server LNA and any other LNAs accessing this data object;

c) any client MAUs connecting to the data object also get a local MCP code. This code, together with the client MAU and the local LNA codes, form an identification triplet for the client's reference to the data object.

The server and client MAUs need only use their local MCP code internally, while the LNAs normally use the complete identification triplet. The server MAU and all its clients will normally have different MCP codes.

NOTE 1 The interface identity is used to reference the interface to which the MCPs belong. It is the MCP identity code that is used during transactions. These must be unique over all interfaces used by a MAU.

NOTE 2 The MCP identification code is local to each MAU. The involved LNAs need to be able to convert between the triplet codes, as indicated with the difference between identity types 2 and 3.

9.5 Interface code (`mcid_m`)

The local MAUs also maintain an interface code of the same format as the MCP code. As the scopes for these codes are different, it is possible for the MAU to use, for example the first MCP code in an interface as the code for that interface.

The LNAs use their MAUs interface codes to identify the interfaces, much as the MCP identification described in the previous subclause.

9.6 Data object name (`mcname_m`, `mciname_m`)

9.7 Interface name (`ifname_m`)

The interface name is a null terminated character string of the type `ifname_m`. It has a length of maximum `IFNAME_LEN` characters excluding null. All characters except null are allowed in the string.

9.8 Session identity (word16_m)

This identity code is local to each LNA and is used to identify an association between two MAUs. The session identity shall be unique and the same for all associations between two MAUs on a single LNA. The session identity shall not be reused if one association is broken and then re-established.

9.9 Transaction identity (`word32_m`)

The transaction identity is used to identify a certain transaction that is requested by a client MAU. The transaction identity may change when the transaction is sent through the client's LNA, to the server's LNA and then to the server. The requirement is that each acknowledgement corresponding to a certain request on a specific link, shall use the same transaction identity as in the original request.

9.10 Format string

The attribute value match mechanism uses a format string to check that a data object's data structures and allowed transaction type match between server and client. The format string consists of a transaction type specification (defined in 9.10.2) and one or two data structure specifications (defined in 9.10.1). The complete format string structure is defined in 9.10.3

The format string is encoded in a null terminated text string. The maximum length of the format string is dependent on the message sizes supported by the T-profile in use.

9.10.1 Data record format encoding

The one or two (for function type) data structures associated with each data object are encoded in a text string. The structure of the string is defined in this subclause.

Table 49 describes the composite data types and how their format sub-string is constructed. The construction of the format string does also define the construction of the composite elements (see 8.4.3.5 for more information on format strings).

Table 49 – Composite element formats

Structure	Format	Short format	Description
Basic type	B	b	Defined as one format code in table 3, for example **i16** or **c8**
Element	E	e	A basic type, a record, an array, union or a var-array
Basic element	X	x	A basic type, a record or union
Array length	N	N	Array length as a decimal number encoded in the text string
Record	**(**E+**)**	**(**e+**)**	A concatenation of format sub-strings in parenthesis. The format specification is read from left to right
Union	**{**B:E+**}**	**{**be+**}**	As for record, but with an additional index as for the var-array
Array	**[**N**]**X	*****Nx	Length in square brackets in front of the format, N is the length of the array in X type entities
Var-array	**[**B**:**N**]**X	*****bNx	Actual length of transmitted array placed in an <u>unsigned</u> entity. B defines type: Only **w8**, **w16** or **w32** are legal. N is the maximum length of array in X type entities
Opaque block	<B**:**N:ID>	**<**bN:ID**>**	Actual number of octets transmitted placed in an <u>unsigned</u> entity. B defines type: Only **w8**, **w16** or **w32** are legal. N is the maximum length of array in octets. ID is the name of the specification that is needed to interpret block. ID is a sequence of printable letters, except the pointed brackets.

The format string for a data object will consist of the concatenation of format code strings defined in Table 3 and the following special characters: '**<**', '**>**', '**(**', '**)**', '**[**', '**]**', '**{**', '**}**', and '**:**'. In addition, all decimal numbers are allowed in array length specifications. The following list defines the four types of data objects that are supported:

a) **basic type**. This is one of the native types as defined by Table 3.

   EXAMPLE – "w16", "f64"

b) **record**. This is a collection of a number of previously defined records, basic types, or arrays.

   EXAMPLE – "(w16(w32i16))"

c) **union**. This is a collection of a number of previously defined records, basic types, or arrays with an additional integer indicating which of the elements that is transmitted.

   EXAMPLE – "{w16:w16(w32i16)}"

d) **fixed length array**. This is a one-dimensional fixed length array of basic types or records. The length of the array (in array type units) is encoded in the format string as a decimal number.

   EXAMPLE – "[32](w16(w32i16))", "[20]w16"

e) **variable length array**. This is a one-dimensional variable length array of basic types or records. The maximum length of the array is encoded in the decimal number. The actual transmitted length of the array will be transmitted in one of the three unsigned integer basic types. The integer type used is encoded in the format string.

   EXAMPLE – "[w8:32](w16(w32i16))", "[w16:300]w32"

f) **opaque block**. This is a sequence of octets that need to be interpreted with the help of some named specification.

EXAMPLE – “<w8:32:ObjectMethod_45>”

It is not legal to declare an array of arrays. Opaque blocks will normally be used as stand-alone types and not embedded in other structures.

9.10.2 Transaction type coding

Each data object supports exactly one transaction type. This transaction type is coded as a single character in the format string. The character codes are defined in Table 50 as the first character in the format column.

9.10.3 Complete format string

The format strings associated with the transaction types are defined in Table 50. The *X*es in the table represent the data object format string as defined in 3.4.3. The bold faced characters represent reserved characters used in the format string.

Dependent on transaction type one or two (for function) data record format strings (*X*es) need to be included in the format string. It is also legal to define data objects with one or two empty data records. In this case, the corresponding data record format strings are empty.

Table 50 – Format string structure

Transaction type	Format	Description
Function	**f/**X1**/**X2	Function, write X1 and read X2
Write	**w/***X*	Write X
Read	**r/***X*	Read X
Subscribe	**s/***X*	Subscribed read X
Individual subscribe	**i/**X1**/**X2	Individual subscribe, write X1, subscribe on X2
Anonymous subscribe	**a/**X	Anonymous subscribe, read or write X
Broadcast subscribe	**b/***X*	Broadcast subscribe on X
Non-acknowledged write	**n/***X*	Non-acknowledged write X

The complete format string will be a null-terminated text string consisting of a leading character specifying the transaction type supported by this data object. Thereafter follows one or two data object structure specifications, each headed with a slash (‘/’).

EXAMPLE – “f/(w16(w32i16)/w32”, “n/(w32w32)”, “f//w16”, “f/w16//”, “w/”

9.11 Password (`password_m`)

The password is a null-terminated text string placed in a `password_m` data type. At most `PASSWORD_LEN` characters excluding the terminating null are significant in the password. All characters except null are legal.

Note that the password is not part of the MCP attributes.

10 Data marshalling

10.1 Introduction

This standard defines a data transport mechanism that will conserve the representation and precision of data entities exchanged between client and server. The transport mechanism will also conserve boundaries between individual data items and aggregates as supported by the standard.

It is the responsibility of the API to convert between its internal data record representation and the representation used in the transport mechanism.

The conversion principle will be different on different computers and programming languages.

a) One cannot guarantee that there is a one to one mapping between the basic data types defined in this standard and the target computer architecture, for example a given computer may not support all basic data types. It is then necessary to define a new mapping that conserves precision and representation.
b) Different computers use different principles for ordering of the most and least significant octets within a larger numeric entity.
c) Different programming languages and operating systems use different principles for ordering and alignment of individual data items within aggregates (arrays and records).
d) Different implementations of a compiler for the same programming language on the same computer architecture may in some cases use different ordering or alignment within aggregates.

For the specification of data object attributes, the structure of the data record will be encoded in a text string (format string). The structure of this string is based on a one to one mapping between data record structure and format string. Thus, it is in principle possible to define a general data marshalling routine that does the necessary conversion based on this format string. This routine must be able to handle the problems outlined above. On the other extreme, some implementations can supply different data marshalling routines for each different data record structure.

10.2 Network octet order

10.2.1 General

The order of transmission of data described in this standard is resolved to the octet level. Multi-octet entities or composite types are sent packed, i.e. with no padding between elements or octets.

The format string will give an unambiguous description of how to pack data into or out of network transmission orders.

10.2.2 Basic types

The `bool_m` type in single entity or as array will be transmitted in whole octet entities. Up to eight `bool_m` entities will be packed into each octet. Least significant bit of the octet represents the lowest index or, if it is a single entity, the entity itself.

NOTE A part of a record with a number of individual bool_m entities will be transmitted as the same number of octets.

The 8-bit character shall be transmitted as its direct numeric representation encoded in an 8-bit unsigned integer (ISO/IEC 8859-1).

The 16-bit character shall be transmitted as the direct representation as two octets, most significant first, as specified in ISO/IEC 10646-1.

All other single octet entities will be sent as single octets. All other multi-octet types shall be transmitted in their specified binary representation with the most significant octet first.

Floating point numbers shall be transmitted as specified in IEEE 754, most significant octet first.

10.2.3 Composite types

Arrays shall be transmitted with the lowest index first. Variable length arrays shall be transmitted with the element count in front of the lowest index element. Zero length variable length arrays are transmitted as length field only.

Records shall be transmitted in the order they are defined in the companion standard. The first element of the definition is transmitted first. The order in the definition is determined by the companion standard.

Unions are transmitted as the index entity first and the indexed element immediately following in the same way as for records.

10.2.4 Messages

Messages are a collection of basic and derived types. The message shall be transmitted as described in the message definition table, line by line, top line first and without any extra (padding) octets.

Some messages include an indeterminate length octet field identified as `word8_m[]` or `string_m`. This field shall be transmitted as defined in 3.1.13 or as a null-terminated text string with the left-most character first (for `string_m`).

The rules in this subclause apply to LNA-LNA messages as well as MAU-LNA messages when the latter is transmitted over an bit-stream communication link.

10.3 Pack and unpack routines

10.3.1 Introduction

This clause assumes that the API contains routines to convert a native representation into the protocol octet order (a pack routine). Likewise, it will also contain routines for conversion between protocol octet order and native order (unpack routines).

These routines may be particular to each different type of data structure, or a general pack/unpack routine may be devised, based on the parsing of the format string.

The API will call the data unpack routine just before handing data over to the MAU. Similarly it will call the data pack routine just after the MAU has created a data output structure, but before the data is committed to the output message queue.

There are several ways to create these routines:

a) one pack/unpack routine for each record type. This is relatively inefficient, but easy to implement and generate automatically;

1 one general pack/unpack routine that uses offset information from a data record compiler. The record compiler would have to be run for each compiler/architecture type. This may cause problems for software that is intended to be ported;

1 one general pack/unpack routine that can use the format string directly. This means that the compiler from the previous item essentially is built into the routine.

10.3.2 Pack routine

This routine will convert MAU internal data record representations to the standard transport format. This routine must be called before a write type request or a read type acknowledgement is sent.

Input:

– format string specification;
– data record.

Output:

– octet string buffer.

Error codes:

– overrun in output buffer;
– syntax errors in format string.

10.3.3 Unpack routine

This routine will convert the standard transport format to MAU internal data record representation. This routine shall be called before a read type acknowledgement or a write type request is processed.

Input:

– format string specification;
– octet string buffer.

Output:

– data record.

Error codes:

– overrun in input buffer;
– syntax errors in format string.

11 Communication link between MAU and LNA

11.1 Introduction

The MAU-LNA communication can be done through host computer local IPC mechanisms or by an explicit communication link transport service. The specific properties of an IPC mechanism are not covered by IEC 61162-4 except to the degree that the mechanism must be able to support the MAU-LNA message exchange covered by this part of the standard.

This clause specifies the general semantics for a general MAU-LNA communication service based on some kind of communication link.

Some T-profile specifications will in some cases define an appropriate MAU-LNA communication protocol. A protocol for use over standard TCP/IP is defined in IEC 61162-410.

11.1.1 General service specification

The MAU-LNA protocol can in principle use any transport service that satisfies the requirements defined in this clause.

The general MAU-LNA protocol transport service shall satisfy the requirements specified by the following subclauses.

11.1.2 Point to point

The service transfers data on a (virtual) point to point communication link between one client (MAU) and one server (LNA).

11.1.3 Connection oriented

The use of the service has three main phases: Establishment of link, transfer of data and closing of the link.

Establishment of the communication link is initiated by the client (MAU). Before the link can be established, it is necessary that the server (LNA) has announced its willingness to accept connection requests from the client. The server listens for connection requests at the announced connection address.

11.1.4 Symmetrical and full duplex

Except for the establishment of the link, all other use of the link is symmetrical, i.e. data transfer and closing the link can be initiated by either client or server at any time. The link shall be full or half duplex.

11.1.5 Message based

An atomic data message sent from one side of the link shall be delivered to the receiver in one atomic operation. The service shall be able to handle a maximum message size suitable for the proposed use (MAU-LNA messages).

11.1.6 Priority

The message link shall support the three priority levels used by this standard to the best possible level. Messages on a higher priority level shall be processed before messages on a lower priority level. Messages within the same priority level shall be processed on a strict first come first served principle.

NOTE Some implementation may not be able to support priority. In particular, the TCP/IP link defined in IEC-61162-420 is intended for wide area network test integration and does not support priority.

11.1.7 Buffering and flow control

The service shall buffer incoming and outgoing messages if these cannot be processed immediately by the service user or the transport layer, respectively. The buffer space shall be sufficiently large to accommodate a reasonable number of messages. Buffering shall be complemented with flow control as is appropriate for the protocol in use.

11.1.8 Reliable transfer

A message sent by this communication service shall be guaranteed to an acceptable level to be delivered with correct contents and in correct sequence to the receiver. A suitable error detection (and optionally error correction) scheme shall be employed. If errors are detected they shall if possible be corrected and an error message shall be delivered to at least one of the service users (client or server).

11.1.9 Error reporting

Both the client and server side of the service provider shall report the communication error to the user of the service. Unrecoverable errors shall be handled by closing the link immediately and notifying the user of the error.

12 General principles for module functionality

12.1 Flexibility in receiving, conservatism in sending

All modules should be flexible in the interpretation of messages they receive. They should not enforce parts of the standard that are of no consequence to them. This will help to cater to possible extensions of the standard and also help to make sub-standard or marginal implementations operate as expected. Some examples are:

a) maximum sizes (message lengths, string lengths, etc.) should not be enforced by receiving modules except where absolutely necessary;
b) the LNA should avoid checking format strings or data fields relating to MAU's data objects. The format strings are not important to the LNA and future expansion of the standard may introduce new format conventions;
c) implementations that can cause deadlocks due to missing or unexpected messages should be avoided. New implementations should be able to receive and process unexpected messages at any time. They should also avoid being deadlocked due to expected, but missing messages;
d) unexpected or unknown messages that are correctly formatted (see next subclause) can in many cases be silently discarded;
e) extra and unrecognized information after known messages should be silently discarded. It can be expected that new versions of the protocol use this mechanism to add functionality to existing messages.

All modules shall adhere strictly to the standard regarding outgoing messages. They should as far as possible implement all optional restrictions described in the standard. This will help to ensure that any sub-standard or marginal implementation of the protocol operates as expected.

NOTE There is a trade-off between being forgivable to marginal implementations and allowing errors to propagate in a system. The examples shown in this clause are informative only and need to be considered carefully at system design time.

12.2 Garbled messages

Each module in the system shall ensure that messages received (and sent) are correctly formatted. Format errors occur when one or more message data fields contain impossible values. Wrong, but plausible name or identity fields shall not generally be regarded as formatting errors. A special magic number is inserted at the front of all messages to make the detection of garbled messages easier.

An incorrectly formatted message should generally cause the receiving module to close any point to point communication link on which the message arrived.

NOTE The trade-off between being flexible in the reception of messages (e.g. silently discard unexpected messages) and detecting bad messages must be considered carefully when new implementations are defined.

12.3 Closed communication links

A remote close of a point to point communication link indicates a communication error. The module detecting the close must immediately tidy up all internal state relating to entities on the other side of the link. It can then try to re-establish the link.

Annex A
(normative)

Message codes

Table 51 is a summary of the message codes used in this standard. The value column defines the numerical value for each code. The definition of the message can be found in clause 8.

Table 51 – Message codes

Symbol	Value	Description
LNA-LNA multi-cast messages		
CC_DEADMAU	0x0401	Warning about death of MAU
CC_DEFMAU	0x0402	Define a new MAU
CC_METOO	0x0403	Warning of duplicate
CC_REQMAU	0x0404	Request a MAU
CC_WATCHDOG	0x0405	General watchdog message from LNAs
LLBC_AACK	0x0406	Anonymous broadcast acknowledgement
LLBC_SACK	0x0407	Broadcast subscribe acknowledgement
LNA-LNA reliable messages		
LL_ALIVE	0x0301	Liveness request
LL_DATA	0x0302	Data transfer
LL_DATAC	0x0303	Data transaction cancel
LL_IFACK	0x0304	Interface connect acknowledge
LL_IFREM	0x0305	Interface removal
LL_IFREQ	0x0306	Interface location request
LL_MAUACK	0x0307	MAU name acknowledge
LL_MAUREQ	0x0308	MAU name request
LL_NOMCP	0x0309	Returned bad transaction
LL_SESSION	0x030a	Session control message
MAU-LNA messages		
MAPI_AACK	0x0201	Anonymous broadcast data
MAPI_AREQ	0x0202	Anonymous broadcast request
MAPI_BACK	0x0203	Broadcast subscribe data
MAPI_BREQ	0x0204	Broadcast subscribe request
MAPI_CACK	0x0205	Cancel acknowledge
MAPI_CREQ	0x0206	Cancel request
MAPI_DACK	0x0207	Individual subscribe data
MAPI_DREQ	0x0208	Individual subscribe request
MAPI_FACK	0x0209	Function call acknowledge
MAPI_FBACK	0x020a	Broadcast subscribe first acknowledge
MAPI_FDACK	0x020b	Individual subscribe first acknowledge
MAPI_FREQ	0x020c	Function request
MAPI_FSACK	0x020d	Subscribe first acknowledge
MAPI_IACK	0x020e	Interface open acknowledge

Symbol	Value	Description
MAPI_IDOWN	0x020f	Signal remote interface down
MAPI_IOACK	0x0211	Interface open acknowledge from server
MAPI_IOPEN	0x0210	Interface remote connect request
MAPI_IREM	0x0213	Destroy interface
MAPI_IREQ	0x0212	Interface connect request
MAPI_NREQ	0x0216	Non-acknowledged write request
MAPI_OACK	0x0214	MAU open acknowledge
MAPI_OREQ	0x0215	MAU open request
MAPI_RACK	0x0217	Read acknowledge
MAPI_RREQ	0x0218	Read request
MAPI_SACK	0x0219	Subscribe acknowledge
MAPI_SESSION	0x021a	Session control
MAPI_SREQ	0x021b	Subscribe request
MAPI_WACK	0x021c	Write acknowledge
MAPI_WREQ	0x021d	Write request
NOTE Although the entities are organized alphabetically and assigned numerical values after that principle, this is not intentional and cannot be relied upon. The numerical values must be taken directly from the table.		

Annex B
(normative)

Error codes and message field values

Table 52 defines the numerical value for status codes and other message field values used in this standard. the final column gives a reference to the first occurrence of the symbol. Note that a symbol may also be used in other contexts, but with basically the same meaning.

Table 52 – Message field values

Symbol	Value	Description	Reference
General message field values			
LM_CONGESTION	13	Not delivered due to congestion	8.2.4.2
LM_OK	0	General positive acknowledge code	8.2.1.5
LM_TIMEOUT	3	User timeout on a request	8.2.1.4
MV4_MAGIC	0x5844	Magic message code for this standard	8.1.1
Status code values (normally some form of error)			
LM_AUTHEN_FAIL	15	Server denied connection	8.3.4.2
LM_CLIENTS	16	Too many clients	8.3.4.2
LM_CONGESTION	13	Not delivered due to congestion	8.2.4.2
LM_DUPLICATE_MAU	20	Second MAU with this name	8.2.2.2
LM_FORMAT_FAIL	4	Data record or transaction type mismatch	8.2.2.2
LM_IF_EXISTS	12	Interface already exists	8.2.3.2
LM_RMAU_DOWN	5	Remote MAU down	8.3.4.2
LM_INVALID_MCP	8	Request with an useless MCP code	8.2.3.2
LM_LIMIT	18	Message cancel because too many transactions	8.2.4.2
LM_MAU_NOT_FOUND	0	Remote LNA did not have named MAU	8.3.3.2
LM_NOTHING_TO_CANCEL	10	Cancel on non-pending transaction	8.2.4.5
LM_NOT_SUPPORTED	7	Protocol version not supported	8.2.2.2
LM_NO_INTERFACE	14	Interface not found	8.3.4.2
LM_NO_SUCH_MCP	9	Request for unknown remote MCP code	8.2.4.2
LM_OK	0	General positive acknowledge code	8.2.1.5
LM_PASSWORD_FAIL	11	Mismatch in interface passwords	8.3.4.2
LM_RMAU_DOWN	5	Remote MAU down	8.3.4.2
Control data object function codes			
M_KILL	3	Force MAU session to end	6.5.2
M_CLOSE	1	Request that MAU end session	6.5.2
M_RESET	2	Request that MAU end session and restart	6.5.2
M_RESTART	5	Internal MAU code to start a new session	6.5.2
M_STATUS	4	Request MAU status	6.5.2
Session control codes			
SESSION_C0	0	Normal operation	5.2.4
SESSION_C1	1	Low priority traffic is stopped	5.2.4
SESSION_C2	2	Normal priority traffic is stopped	5.2.4
SESSION_DOWN	3	Session down	5.2.4

Annex C
(normative)

Symbolic constants

Table 53 defines the symbolic constants (except for message and error codes) used in messages in this standard. Message codes were defined in annex A, error codes in annex B. The reference column shows where the symbolic constant is described.

Table 53 – Symbolic constants

Symbol	Value	Description	Reference
Various text string lengths			
MV4_MES_MIN	512	Minimum message size	4.5.4
IFNAME_LEN	31	Length of interface name string ex. terminating null	9.7
MAUNAME_LEN	31	Length of MAU name string ex. terminating null	9.5
MCNAME_LEN	31	Length of interface MCP name string ex. terminating null	9.6
PASSWORD_LEN	7	Length of password string ex. terminating null	9.11
Interface and MCP attributes (see 5.4.3 or 5.4.5)			
M_CAN_CONFIGURE	Value defined by API	TRUE if remote configurable allowed	See reference in header
M_CLIENTS		Number of clients	
M_CONN_HANDLER		Connection handler	
M_CONN_RCODE		User return-code for connection handler	
M_DESC		A description string for the interface	
M_FORMAT		The format string	
M_IFCNAME		The name of the class of the interface	
M_IFNAME		The symbolic interface name	
M_INTERFACE		Return interface reference or NULL	
M_IS_ACCEPT		TRUE if accept type	
M_LASTERROR		Last error code	
M_MAUNAME		The symbolic server MAU name	
M_MCPNAME		TRUE if remotely configurable	
M_PASSWORD		The interface password	
M_PRIORITY		Default priority for transactions	
M_RECV_PACK		Unpack-function (marshalling for receive)	
M_RECV_STORE		Input data-buffer	
M_SEND_PACK		Pack-function (marshalling for send)	
M_SEND_STORE		Output data-buffer	
M_SESSION		Session ID for current operation	
M_TIMEOUT		Connection timeout (ms)	
M_TRANS_HANDLER		Transaction-handler	
M_TRANS_RCODE		User return-code for transaction handler	
M_TRANSQUEUE		Length of transaction queue	
Various time related constants			
POLL_INTERVAL	1000	Interval (ms) for LNA poll of MAU	6.8
RETRY_INTERVAL	1000	Interval (ms) for LNA search for MAU	6.2.3

Symbol	Value	Description	Reference
WATCHDOG_INTERVAL	5000	Interval (ms) for transport level watchdog timeout	6.2.3
Various event codes			
EV_IF_CDOWN	Value defined by API	Client disconnects from accept interface	5.5.5
EV_IF_COPEN		Client requests connection to accept interface	5.5.5
EV_IF_DOWN		Interface closed from remote side	5.5.5
EV_IF_NOT_FOUND		Connection timed out or was denied due to,for example high load	5.5.5
EV_IF_NO_CONNECT		Attribute values inhibit the establishment of the interface	5.5.5
EV_IF_OPEN		Interface open	5.5.5
EV_LNA_CLOSE		Request MAU close	5.2.4
EV_LNA_CONNECT		MAU-LNA connection state change	5.2.4
EV_LNA_CONNECTION_ DOWN		Communication link down, state change	5.2.4
EV_LNA_KILL		Force close of MAU.	5.2.4
EV_LNA_NO_CONNECT		Connection failed due to timeout or missing LNA	5.2.4
EV_LNA_OPEN_ERROR		Port description parameter error	5.2.4
EV_LNA_RESET		Internal MAU event: MAU is reset	5.2.4
EV_LNA_RESTART		Request MAU reset	5.2.4
EV_LNA_SESSION		Remote client MAU session changes state	5.2.4
EV_LNA_STATUS		Return MAU status	5.2.4
EV_TRANS_ACK		Transaction completed	5.6.1
EV_TRANS_CANCEL		Transaction cancelled successfully	5.6.1
EV_TRANS_FAILED		Transaction failed (with error)	5.6.1
EV_TRANS_REQ		Transaction request to server	5.6.3
EV_TRANS_SACK		Transaction subscription acknowledge	5.6.3
Other miscellaneous definitions			
OPEN_MCP_ID	0	Pseudo-MCP for MAU control from LNA	6.5.1
PFC_FULL	3	MAU with configuration management capabilities	8.2.2.1
PFC_LNA	1	LNA MAU class MAU	8.2.2.1
PFC_NONE	0	MAU has no PFC related capabilities	8.2.2.1
PFC_SIMPLE	2	MAU has version control capabilities	8.2.2.1
PT_LOW	0	Low priority	4.5.2
PT_NORMAL	0x100	Normal priority	4.5.2
PT_URGENT	0x800	Urgent priority	4.5.2

Annex D
(informative)

Compatibility between MiTS and the IEC 61162-400 series

This annex gives an overview of the incompatibilities between MiTS and the IEC 61162-400 series. This informative annex is included to enable easier migration between the two protocols and also encourage deployment of this new international standard in older equipment.

D.1 General compatibility

MiTS and the IEC 61162-400 series are only compatible on the module level, i.e. applications using the two protocols can coexist on the same computer and network, but they cannot communicate between each other. To establish a data exchange between them, a gateway MAU must be used. This could, for example be built using the general alarm and monitoring companion standard IEC 61162-420.

D.2 MCPs and interfaces

The MCP structure from MiTS is more or less retained, except from some small functional enhancements in this standard that are discussed below. The main difference is that MCPs are now organized in interfaces which are connected *en block*. This means that MCPs do not have individual connection handlers.

Session management: It is now possible to do simple flow control on a MAU to MAU (or MAU to LNA) basis through session control messages. It is also possible to identify to which session (a MAU-MAU association) a transaction belongs (session identifier). This is useful for client authentication.

Null MCP: It is only usable locally for the MAU. The MAU control function is now handled by LNA-MAU (see IEC 61162-420). This is an effect of the introduction of interfaces.

No MAU password: The removal of the full functionality of the null MCP voids the need for a MAU password.

D.3 Transactions

Multiple and delayed transactions: Transactions are now (V4) first class objects that do not have to be acknowledged immediately and which can be issued in any number even though outstanding acknowledgements exist.

New transaction types: Individual subscribe allows clients to set up individual subscription requests with one client. This simplifies the case where different clients need different information. Anonymous broadcast subscriptions make it possible to receive messages from unknown senders. This can be used, for example for interrogating the system about certain information items.

D.4 Data types

New basic types: 64-bit integers have been introduced together with one bit, boolean types. 16-bit characters have also been introduced with representation according to new ISO standards.

New aggregates: Type-safe unions have been introduced together with opaque blocks. The latter makes it possible to send any type of data in a format that is only agreed upon between sender and receiver.

New derived types: The network address and network node numbers have been introduced as new derived data types.

___